Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROYECTO FINAL DE CARRERA

TÍTULO: MONTAJE, CALIBRACIÓN Y PROGRAMACIÓN DEL ROBOT SR1

AUTORES: Javier Salvador Marco
Joan Martínez Recasens

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones Esp. Sist. Electrónicos

DIRECTOR: Spartacus Gomariz Castro

DEPARTAMENTO: Ingeniería Electrónica

FECHA: 13 de Julio del 2007

TÍTULO: **MONTAJE, CALIBRACIÓN Y PROGRAMACIÓN DEL ROBOT SR1**

APELLIDOS: **Salvador Marco** **NOMBRE:** **Javier**

APELLIDOS: **Martínez Recasens** **NOMBRE:** **Joan**

TITULACIÓN: **Ingeniería Técnica de Telecomunicaciones**

ESPECIALIDAD: **Sistemas Electrónicos** **PLAN:** **75**

DIRECTOR: **Spartacus Gomariz Castro**

DEPARTAMENTO: **Ingeniería Electrónica**

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Este proyecto tiene en cuenta aspectos medioambientales: □ Sí □ No

PROYECTO FINAL DE CARRERA

RESUMEN (máximo 50 líneas)

El proyecto se basa en el montaje, calibración y sobre todo programación de un robot SR1.
El primer paso consiste en soldar todos los componentes a la placa, comprobando su correcto funcionamiento, y acto seguido, montar los servos con el chasis y la cámara.
Una vez completado todo el robot, se pasa a la fase de calibración, donde se calibran los cuatro servomotores del robot, la brújula, el sensor de inclinación, los ultrasonidos y los fotodiodos.
Cuando el robot está 100% operativo, se pasa a la parte de programación, donde destacan los 3 programas principales en los que se basa el proyecto.
Antes de comentar los programas propios, se prueba el funcionamiento de los programas que vienen de serie con el robot, como el programa para esquivar obstáculos con el sensor de infrarrojos, el programa para buscar luz mediante células fotoeléctricas, o el programa seguidor de líneas, que hace que el robot siga un circuito o recorrido mediante el sensor de seguidor de líneas.
Los programas propios son los siguientes:

-El primero es un programa para dirigir el robot SR1 mediante un control remoto compatible con Sony.

-El segundo es una modificación del programa Explorer, que se basa en un programa de PC con el que el usuario puede dirigir y recibir datos de nuestro robot SR1 mediante el puerto serie. Como es la temperatura, la posición, la iluminación, las imágenes de la cámara y efectuar un barrido con los ultrasonidos a diferentes niveles.

-El tercero de los programas, se basa en mantener una ruta, dirección norte en éste caso, hasta que encuentra un obstáculo. En ese momento lo bordea, toma una imagen con la cámara y vuelve a reorientar el norte volviendo a dirigirse hacia él.

Palabras clave (máximo 10):

Robot	Ultrasonidos	Brújula	Sensor IR
Servos	Seguidor de líneas	Cámara	BasicX
Sensor de inclinación	Control remoto		

ÍNDICE

1 Introducción

Mediante el gran número de características que tiene el robot SR1, que consiste principalmente en un robot preparado para la exploración. Se puede llegar a conseguir un amplio abanico de posibilidades, con las que se puede sacar un buen rendimiento del robot. Como por ejemplo con la gran cantidad de sensores con los que está formado, puede realizar una ruta evitando cualquier tipo de obstáculo que se encuentre frente a él.

El proyecto se divide en dos partes:

Hardware:

- El montaje de la placa del circuito, donde van depositados los sensores, el microprocesador y los dispositivos de salida. Una vez realizada la placa se observa con detenimiento si no hay ningún error a la hora de soldar y si las pistas son las correctas.

- El montaje del chasis, es donde se ubica la placa del circuito. Una vez finalizado el montaje del chasis, se puede tener en consideración añadirle unas ampliaciones. En el caso de este proyecto se tiene en cuenta la ampliación de la cámara colocando el sensor de ultrasonidos en la torreta de la cámara, también se decide colocar un seguidor de líneas en la parte trasera del chasis.

En cuanto se tiene listo tanto el montaje de la placa, como el montaje del chasis, se pasa a la calibración del robot, en esta parte se tiene que trabajar con la parte de software.

Software:

La calibración consiste en la calibración de los servos de tracción, los servos encargados del movimiento de la cámara y la brújula digital. La calibración se realiza mediante los programas propios del fabricante, aunque como se puede apreciar en el apartado 3.6, se modifican algunos programas para poder calibrar correctamente el robot. Los programas realizados por nosotros, donde se encuentran dos programas realizados desde cero y un programa modificado y optimizado. Los dos programas realizados desde cero se llaman **Dirección Norte** y **Control remoto SONYIR**, el primer programa consiste en realizar una ruta específica siguiendo un rumbo determinado, mientras avanza, se encuentra obstáculos que debe evitar. Cuando se encuentra con el obstáculo, lo filma con la cámara incorporada y lo evita, una vez evitado, reorienta el robot hacia el rumbo determinado. El programa **Control remoto SONYIR**, se realiza el control del robot mediante un mando a distancia, con el que se puede dirigir el robot.

El programa modificado, se llama **EXPLORER**, que consiste en el control por el usuario del robot mediante un programa de PC. La conexión del robot se realiza vía puerto serie, de esta manera, el usuario puede dirigir y recibir datos del robot.

2 Objetivos del proyecto

Los objetivos del proyecto son los siguientes.

1. Conseguir manejar perfectamente el entorno de programación BasicX, comprendiendo los programas hechos por el fabricante y mejorándolos si se puede.

2. La perfecta comprensión de todos los sensores del robot, así como su calibración y su puesta a punto.

3. La realización de programas propios, intentando sacar el máximo partido a los sensores, a la cámara y a los servos del robot

4. Conseguir que el robot sea lo suficientemente inteligente para seguir unas coordenadas sin desviarse de ellas, esquivando todos los obstáculos que encuentre en el camino y mostrándonos una imagen de estos en una televisión u ordenador mediante la micro cámara que tiene colocada en su parte superior.

5. Conseguir que, mediante un mando a distancia Sony, el usuario pueda darle ordenes al robot y hacer que este pueda realizar todos sus movimientos y realizar todas las funciones que no necesiten ser visualizadas en la pantalla de un ordenador, a excepción de la cámara de video, la cual, al llevar un receptor de señal independiente puede ser usada sin problemas.

6. Conseguir solucionar los errores de los programas interesantes del robot, como por ejemplo el programa **"Sr1 Explorer"**, el cual, gracias a la perfecta comprensión del entorno de programación, se puede hacer funcionar correctamente.

3. Montaje del Robot SR1

El robot SR1, figura 3.1, consiste principalmente en un robot preparado para la exploración. Gracias a su brújula y el sensor de ultrasonidos, se consigue que siga una ruta, evitando cualquier tipo de obstáculo que se encuentre frente a él. Se consigue, mediante su receptor de infrarrojos, un control del robot a partir de un mando a distancia Sony; pudiendo mover y realizar las acciones requeridas en el robot a través de él. Otra variedad de control sobre el robot es mediante el ordenador. Con el programa denominado SR1 Explorer, se consiguen realizar las acciones requeridas en el robot a través de los comandos del programa, vía cable de puerto serie.

Para explicar el montaje del robot, se estructura el tema en los siguientes apartados:

- 3.1 Montaje de la placa. En este apartado se explica cómo se monta el circuito y que forma parte de él, sus sensores, sus dispositivos, su alimentación y el microprocesador que constituye el robot.

- 3.2 Descripción de los sensores. En este apartado se explican cuidadosamente de los sensores que forman el robot, con los que el robot puede obtener información del exterior.

- 3.3 Descripción de los dispositivos de salida. En este apartado se explican los dispositivos de salida con los que el usuario recibe información del robot.

- 3.4 Montaje del chasis. En este apartado se describe el montaje del chasis y la manipulación de los servomotores de tracción, que son con los que se moverá el robot hacia adelante, atrás, derecha e izquierda.

- 3.5 Posibles ampliaciones del robot. En este apartado se describen posibles ampliaciones que se le pueden aplicar al robot, como por ejemplo la ampliación de la cámara e incluso si se quiere, cambiar las ruedas por un juego de orugas para que pueda circular por todas las superficies sin ningún inconveniente.

- 3.6 Calibrado. En este apartado se describen los pasos que se siguieron para poder dejar a punto al robot para su utilización.

Fig. 3.1 *Robot SR1*

3.1 Montaje de la placa

La placa principal del robot es una placa PCB de doble cara con pistas por ambas caras, en una de las dos caras es donde se ubican los componentes a soldar. En la placa se pueden encontrar los siguientes dispositivos:

- El microprocesador BASICX 24, que se puede observar en la figura 3.2, es con el que programamos el robot vía puerto serie, es el encargado de ejecutar los programas y realizar todas las funciones que controlan el robot.

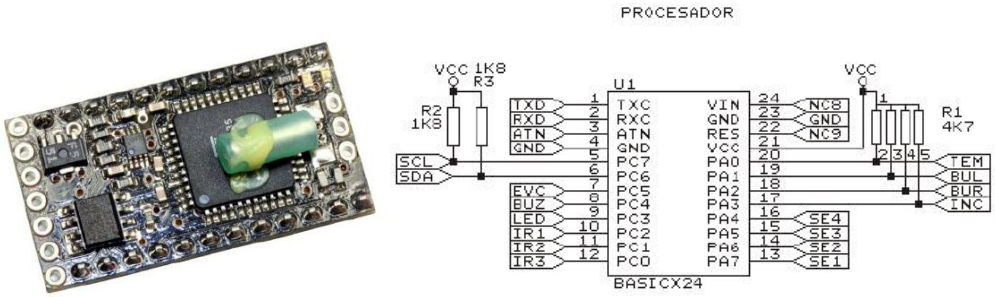


Fig. 3.2 *Microprocesador BASICX 24, encapsulado y su patillaje*

- La conexión para la transmisión de datos al PC mediante puerto serie y vía radio. De esto se habla en el apartado 3.3. Dispositivos de Salida.
- La alimentación de los circuitos, de los servomotores y de la cámara wireless.
- El sistema de posicionamiento, mediante un inclinómetro para decir si el robot está inclinado o no, y una brújula digital con la que se puede saber en qué rumbo está dirigido.
- Los sensores con los que se puede hacer que el robot pueda circular sin tener ningún problema y pueda evitar colisiones y variar el rumbo.

En la figura 3.3 se puede observar cómo queda la placa una vez realizado todo el procedimiento de colocación de componentes y de soldadura.

Fig. 3.1.3 Placa completamente montada

La explicación del montaje de la placa se encuentra detallada en el manual de instrucciones[1] del robot SR1, no obstante se explica a continuación los puntos más conflictivos en el montaje de la placa.

Lo primero que se tiene que hacer es observar las placas e identificar el recorrido de las pistas, por si hay algún cruce indebido, alguna pista cortada y sobretodo hay que reconocer cual es la pista donde circulan la tensión de alimentación y la masa del circuito, esto se puede apreciar en la figura 3.4. De este modo, una vez identificadas las rutas de las pistas y comprobar que todo está correcto, procederemos a identificar los componentes que se han de soldar en sus respectivas ubicaciones. De esta manera es más fácil a la hora de hacer el montaje de los componentes.

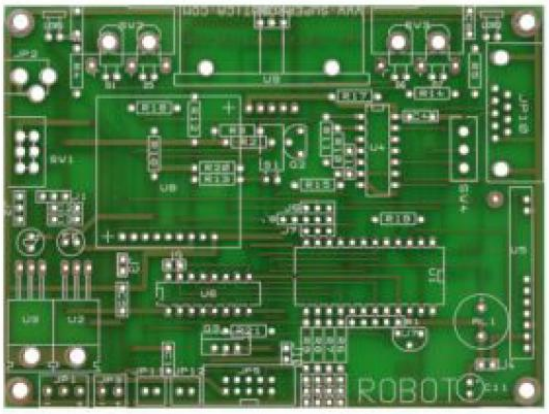


Fig. 3.4 *Placa PCB*

- Uno de los casos es la brújula digital ya que hay que ponerla sin ningún tipo de desnivel, y tiene que quedar totalmente horizontal. Sucede lo mismo para el sensor de inclinación, también tiene que estar horizontal sino puede dar lugar a algún tipo de error en la medición e incluso en el funcionamiento de los programas.

- Se tiene que ir con cuidado con los termorretráctiles de los infrarrojos ya que si se ponen mal, pueden dar lugar a posibles mediciones erróneas.

- Cuando se sitúan los ultrasonidos en la parte superior del chasis junto a la antena, se tienen que quitar ciertas piezas montadas anteriormente, ya que al estar a la vista de los ultrasonidos, provocan que éstos obtengan unos datos erróneos. Con la cual cosa se tiene que quitar el PVC amarillo, que está situado en la parte delantera, y apartar un poco la antena. Para así obtener buenos resultados y no se produzca ningún error a la hora de evitar algún obstáculo.

3.2 Descripción de los sensores

El robot contiene varios sensores con los que se basa para su movimiento. Hay sensores montados en el propio circuito y hay otros como el seguidor de líneas que está montado en la parte trasera del chasis. Se procede a enumerar los sensores que se requieren para la construcción del robot, para luego explicarlos detalladamente.

- 2 Sensores de contacto
- 1 Sensor de inclinación
- 2 Sensores de luz
- 1 Sensor de infrarrojos
- 1 Sensor de ultrasonidos que incluye un sensor de luz central
- 1 Sensor de temperatura digital
- 1 Sensor de brújula digital
- 1 Seguidor de líneas

3.2.1 Sensores de contacto

Los sensores de contacto están formados por dos micro interruptores, también llamados bumpers, tipo final de carrera. Están situados en la parte frontal del circuito de forma que cuando el robot colisiona contra un obstáculo, éste lo detecte mediante la presión ejercida en los paragolpes y estos presionan a la vez los bumpers. Esto se puede apreciar en la figura 3.5 donde se visualiza uno de los dos micro interruptores.

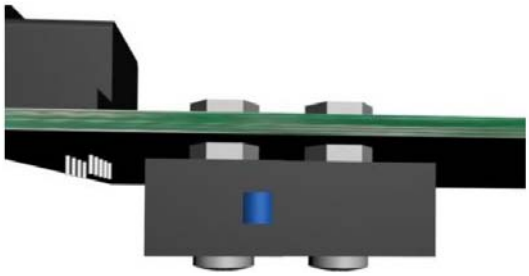


Fig. 3.5 *Micro interruptor*

En la figura 3.6, se muestra que cuando se presionan los bumpers, éstos avisan al robot de que han encontrado un obstáculo y por lo tanto el robot hará el giro convenientemente. La información le llega al microprocesador por la patilla 18 y la patilla 19 donde se encuentran una resistencia en serie, para limitar la corriente que circula y así poder evitar daños con el bumper derecho y en la otra patilla lo mismo para el otro bumper izquierdo.

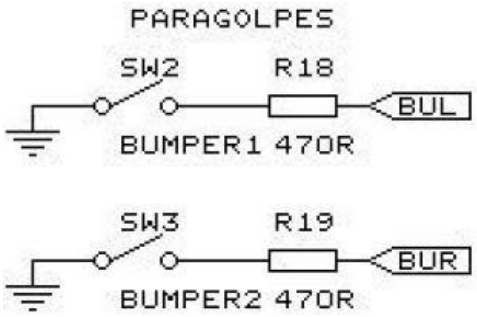


Fig. 3.6 *Esquema de conexión*

3.2.2 Sensor de inclinación

El sensor de inclinación es un interruptor que se cierra en el momento en que se sobrepasa el nivel de inclinación establecido. El interruptor se activa cuando el robot se inclina unos 15 grados hacia arriba por la parte delantera. En la figura 3.7 se puede observar cómo está situado en la placa de circuito. Se recuerda que el sensor de inclinación está totalmente horizontal, para que no haya ningún tipo de error a la hora de hacer las mediciones. El sensor lo que hace es poner a 0 la patilla del procesador cuando se cierra el contacto, por lo tanto, el robot se encuentra con un obstáculo, que hace que el robot se levante y como consecuencia, el sensor de inclinación se activa. Normalmente la patilla está a 1, por lo tanto está en modo entrada y el robot está en posición horizontal.

Fig. 3.7 *Sensor de inclinación*

Como se puede observar en la figura 3.8, se muestra un esquema eléctrico donde va conectado el sensor de inclinación. Igual que con el sensor de contacto, en este también se le conecta una resistencia en serie para limitar la corriente que circula y así evitar daños.

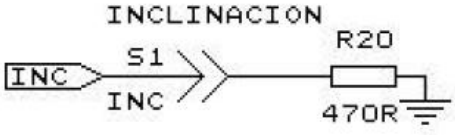


Fig. 3.8 *Esquema de conexión*

3.2.3 Sensores de luz

El robot tiene dos sensores de luz montados en los extremos de la parte delantera. Los dos sensores están formados por fotorresistencias de sulfuro. Éstas cambian de valor en función de la luz que reciben. En la figura 3.9 se visualiza donde están situadas las fotorresistencias y la figura 3.10 muestra el esquema de conexión del sensor.

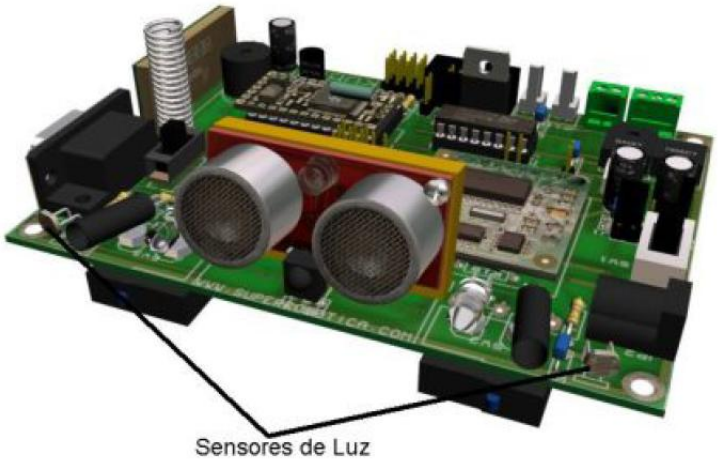


Fig. 3.9 *Sensores de luz*

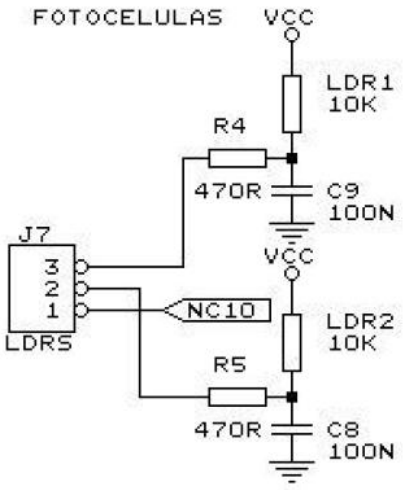


Fig. 3.10 *Esquema de conexión*

Un problema que se ve con el sensor de luz, se da en el programa navegación buscando la luz. El problema consiste en que el robot cuando va buscando la luz, el haz de luz debe incidir prácticamente directamente a las fotorresistencias, cuando no recibe bien la luz, el robot no responde correctamente y no cumple con su cometido.

3.2.4 Sensor de infrarrojos

El sensor de infrarrojos es un circuito integrado que lleva en su interior un sensor de luz infrarroja junto con un filtro y un oscilador de 38 KHz, esto hace que el sensor sea muy sensible a las señales que están moduladas en esta frecuencia. En la figura 3.11 se muestra donde está situado el sensor de infrarrojos en el circuito y en la figura 3.12 se muestra el esquema de conexionado.

Con el sensor de infrarrojos, el robot tiene que recibir la señal luminosa y compararla, para así saber si se encuentra un obstáculo delante de él o no, y si se da el caso que se encuentre algún obstáculo, evitarlo.

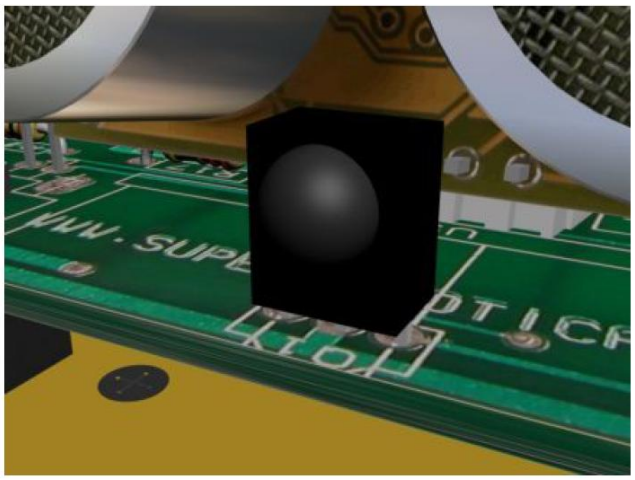


Fig. 3.11 *Sensor de infrarrojos*

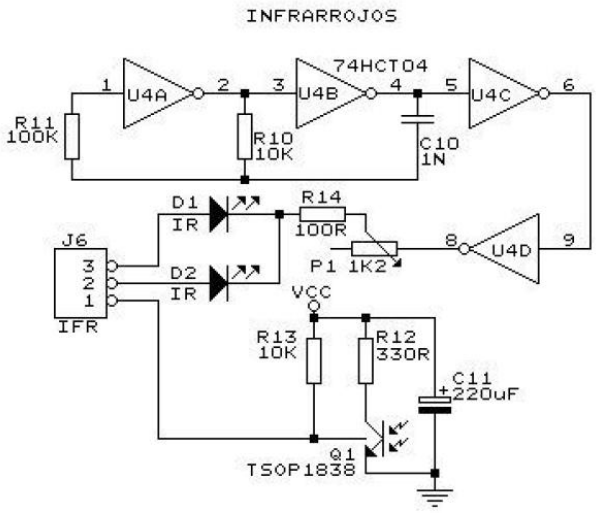


Fig. 3.12 *Esquema de conexión*

El problema que se tiene con el sensor de infrarrojos, es que para espacios reducidos no sirve, ya que tiene un gran alcance y siempre está detectando obstáculos, siendo así un sistema un tanto inexacto.

3.2.5 Sensor de ultrasonidos

El sensor de ultrasonidos utilizado en el robot, figura 3.13 es el modelo SRF08 de Devantech. El sensor es un completo modulo electrónico que incluye su propio procesador para hacer el cálculo de la distancia, junto con el emisor y el receptor de ultrasonidos. Consiste en mandar una serie de impulsos ultrasónicos por el aire y luego medir el tiempo que se tarda en recibir el eco cuando el sonido rebota en algún objeto. Una vez el sonido es recibido por el ultrasonidos, el robot tiene programada cierta distancia para evitar el objeto, si la distancia medida por el sensor es menor a la distancia programada el robot debe evitar el obstáculo. Si la distancia medida es superior a la distancia programada el robot puede continuar por el mismo camino.

Fig. 3.13 *Sensor de ultrasonidos*

El modelo de este sensor de ultrasonidos aparte de tener los ultrasonidos, tiene en el centro una célula fotoeléctrica que nos facilita información acerca del nivel de luz.

3.2.6 Sensor de temperatura

El sensor de temperatura está formado por un sensor de tipo DS1820 de Dallas Semiconductor, figura 3.14. Este sensor es un completo termómetro digital cuya característica más significativa es que se comunica con el procesador usando un protocolo de un hilo. Como su nombre indica la función que tiene el sensor de temperatura en el robot, es la de ofrecer la temperatura ambiental que hay en el recinto.

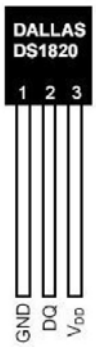


Fig. 3.14 *Sensor de temperatura*

La figura 3.15 muestra el esquema de conexión del sensor de temperatura.

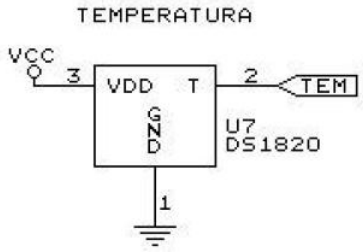


Fig. 3.15 *Esquema de conexión*

3.2.7 Sensor brújula digital

El sensor de brújula digital es un sensor de campos magnéticos formado por el módulo CMP03 de Devantech. La función principal de la brújula es la de determinar el rumbo, en grados, en el que el robot se encuentra, para así poder trazar rumbos o incluso para hacer un sistema de posicionamiento y determinar en donde se encuentra el robot. La figura 3.16 muestra el sensor de la brújula digital. La figura 3.17 muestra el patillaje de la brújula.

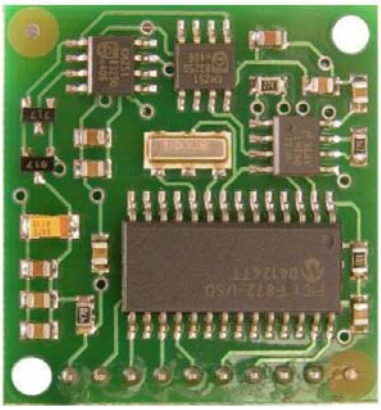


Fig. 3.16 *Sensor Brújula digital*

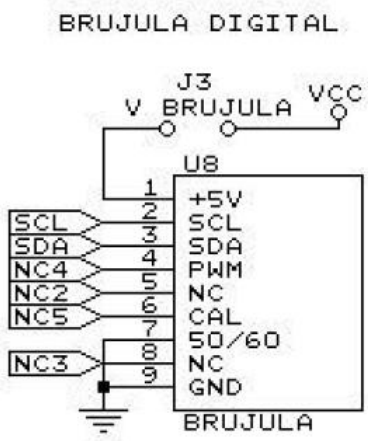


Fig. 3.17 *Esquema de conexión*

3.2.8 Sensor seguidor de líneas

El sensor seguidor de líneas es un modulo electrónico de Lynxmotion que se coloca en la parte posterior del robot. Está formado por tres emisores y detectores de infrarrojos separados 25 mm entre sí. El sensor también está formado por tres leds, que lo que hacen es facilitar visualmente que infrarrojo es el que está detectando o no la línea. Se puede apreciar en la imagen 3.18 de que está formado el sensor.

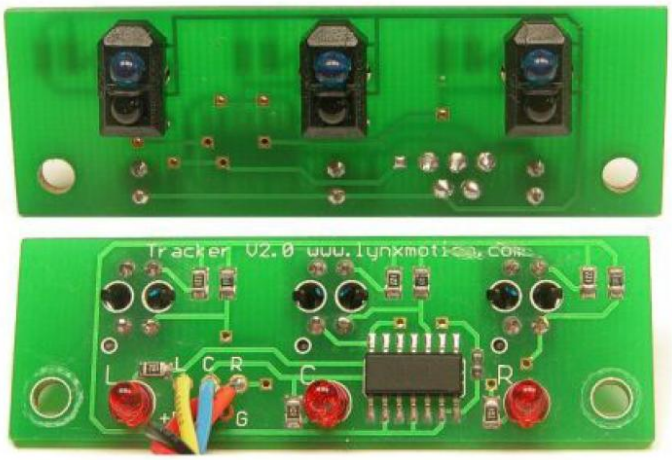


Fig. 3.18 *Sensor seguidor de líneas*

La función principal del seguidor de líneas es la de detectar mediante los emisores y detectores de infrarrojos si debajo del robot hay algún tipo de línea. Si es así el robot sigue dicha línea intentando no perder su rastro. Si pierde el rastro, el robot debe de intentar volver lo antes posible a la línea para hacer el recorrido en el menor tiempo.

La figura 3.19 muestra el esquema de conexiones del sensor seguido de líneas.

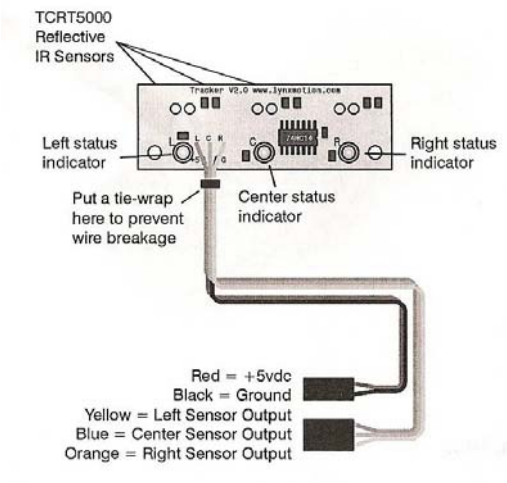


Fig. 3.19 *Esquema de conexión*

3.3 Descripción de los dispositivos de salida

El robot está formado por varios dispositivos de salida con los que puede actuar con el ámbito que le rodea en respuesta a la información obtenida por los sensores. Se procede a enumerar los dispositivos de salida que se requieren para la construcción del robot, para luego explicarlos detalladamente.

- 1 Zumbador piezoeléctrico
- 2 Emisores de infrarrojos
- 2 Diodos led de alta potencia
- 2 Leds rojo y verde de señalización en el chasis mas 4 en la ampliación de la cámara
- 4 Conexiones para servomotores
- 1 Controlador para motores DC
- 1 Puerto serie RS232
- 1 Transceptor inalámbrico
- 1 Conector para periféricos

3.3.1 Zumbador piezoeléctrico

El zumbador piezoeléctrico, figura 3.20, es capaz de generar tonos de señal de diferentes frecuencias que se emplean como señales de aviso, alarmas e indicadores de estado. El dispositivo al conectarle electricidad, hace que vibre y en consecuencia genera un sonido. Su frecuencia de resonancia es de 2000Hz. En la figura 3.21, se muestra el conexionado del altavoz.

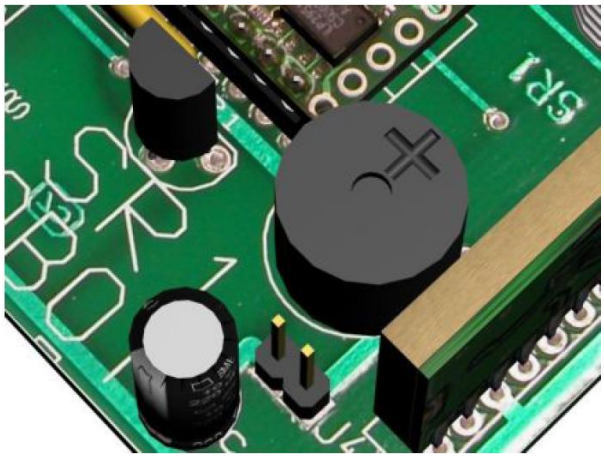


Fig. 3.20 *Zumbador piezoeléctrico*

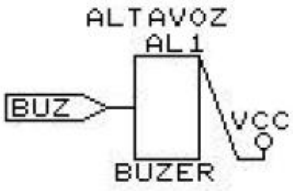


Fig. 3.21 *Esquema de conexión*

3.3.2 Emisores de infrarrojos

Los sensores emisores de infrarrojos permiten enviar ráfagas de infrarrojos en cada uno de los dos leds de forma independiente, ya que están conectados en las patillas 10 y 11 del microprocesador. En la figura 3.22 se ven donde están situados los emisores. Con los emisores podremos enviar las ráfagas para ver si hay algún obstáculo delante del robot, y entonces el sensor de infrarrojos pueda recibir el retorno de la ráfaga después de haber sido rebotado por el obstáculo.

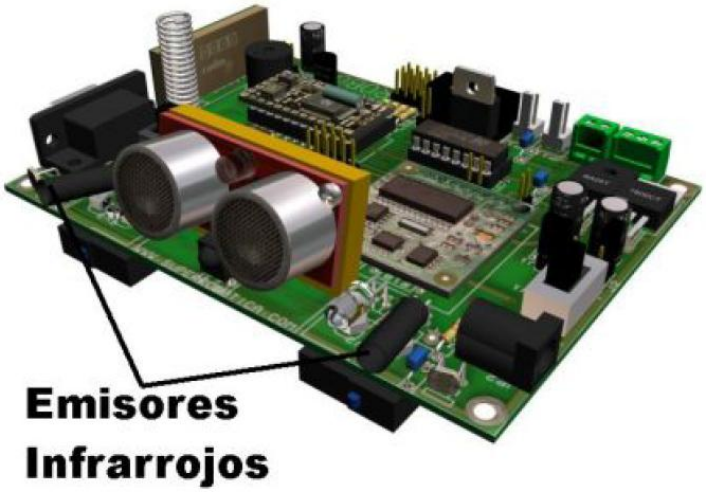


Fig. 3.22 *Emisores de infrarrojos*

3.3.3 Emisores led de alta potencia

Se montan dos leds emisores de alta potencia en la parte frontal del robot y cuatro leds más, estos últimos están colocados en la parte de la ampliación de la cámara como se explicará en el apartado 3.5. En la figura 3.23 se puede observar cómo están montados los diodos leds.

Fig. 3.23 *Emisores led de alta potencia*

Su función es la de hacer señales suficientemente llamativas, para que se vean a plena luz del día y a una distancia de varios metros. Cuando no hay luz sirven para hacer de foco para el robot y así iluminar su camino. La figura 3.24 muestra el esquema de conexiones de los leds de alta potencia situados en la placa.

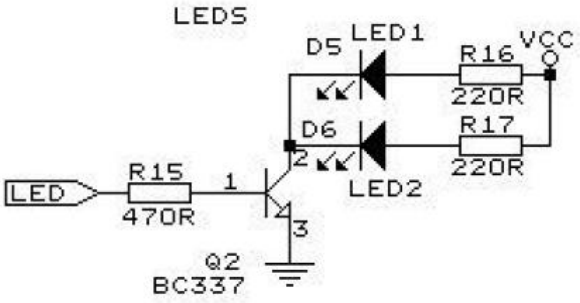


Fig. 3.24 *Esquema de conexión*

3.3.4 Leds rojo y verde de señalización

Los leds rojo y verde están incorporados al microprocesador y son muy útiles para señalizar estados e indicar modos de funcionamiento del robot.

3.3.5 Conexiones de Servomotores

Los servomotores son unos dispositivos que incluyen un pequeño motor eléctrico, una caja reductora y un circuito electrónico que permite controlar al motor. En la figura 3.25 se puede observar donde ha de ir conectado cada servomotor.

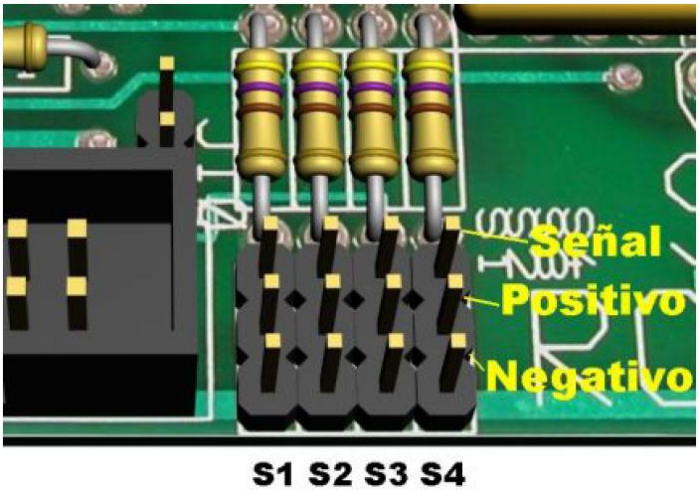


Fig. 3.25 *Conexiones de Servomotores*

En nuestro caso se utilizan dos servomotores para la tracción del robot, un servomotor en la ampliación de la cámara para mover horizontalmente el soporte y el último para poder mover el soporte de la cámara en vertical. En la figura 3.26 tenemos el esquema de conexión de los servomotores.

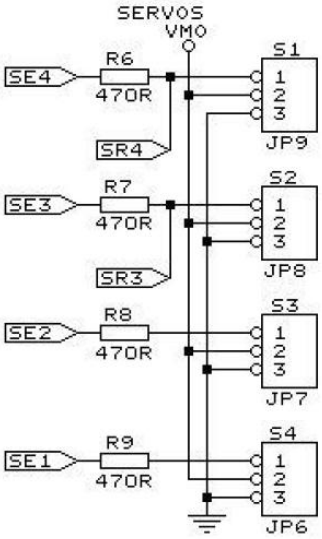


Fig. 3.26 *Esquema de conexión*

La conexión de los servomotores en la placa tiene que testearse para saber donde están situadas masa, alimentación y la señal, para no estropear los servomotores. Esto se tiene que hacer ya que en ninguna parte dice como deben de ir colocados los conectores de los servomotores en la placa del circuito.

3.3.6 Controlador de motores DC

El controlador de motores DC permite conectar dos motores de corriente continua para utilizarlos en sustitución de los servos. En la figura 3.27 se muestra donde van conectados los motores DC. La figura 3.28 muestra el esquema de conexión para los motores DC.

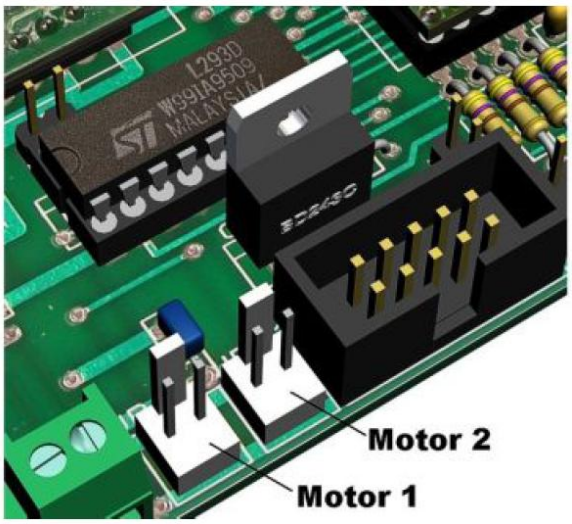


Fig. 3.27 *Conexiones de motores DC*

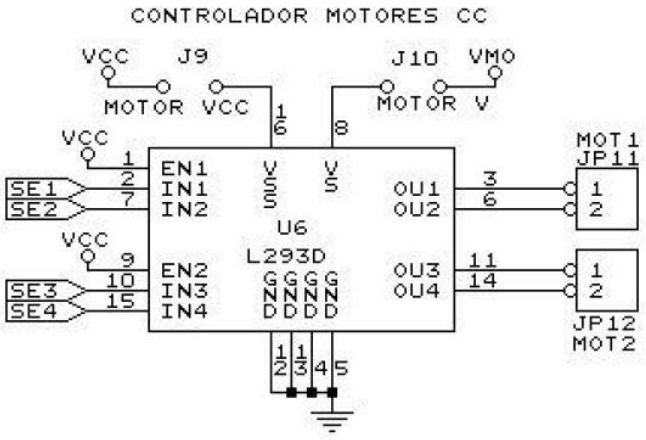


Fig. 3.28 *Esquema de conexión*

3.3.7 Puerto Serie RS232

El puerto serie, figura 3.29, sirve para descargar los programas desde el PC al microprocesador del robot, también sirve para tener una comunicación bidireccional con el ordenador, tanto para recibir datos como para enviarlos. De esta manera se consigue tener un control del robot mediante instrucciones enviadas desde el ordenador, que está conectado al robot mediante el puerto serie.

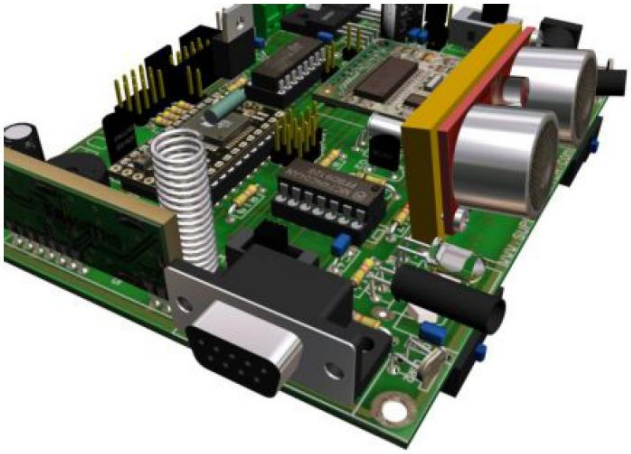


Fig. 3.29 *Puerto Serie RS232*

En la figura 3.30 se especifica el patillaje del conector del puerto serie.

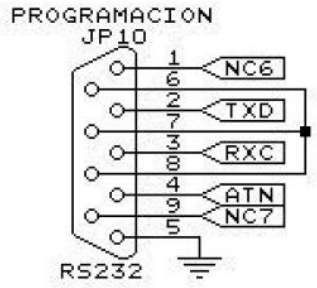


Fig. 3.30 *Patillaje del conector puerto serie*

3.3.8 Transceptor inalámbrico

El transceptor inalámbrico, figura 3.31, es un radio modem capaz de enviar y recibir señales a 19.200 baudios con un rango de unos 50 metros en interior. No se utiliza en ningún momento este tipo de conexión con el ordenador, ya que no se dispone del transceptor inalámbrico de USB para el ordenador. La figura 3.32 muestra el esquema de conexión del transceptor inalámbrico.

Fig. 3.31 *Transceptor inalámbrico*

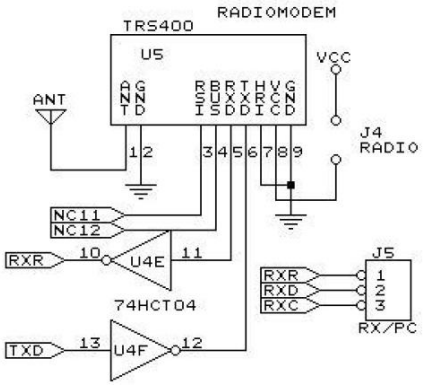


Fig. 3.32 *Esquema de conexión*

3.3.9 Conector de Expansión

El conector de expansión, figura 3.33, permite conectar otros dispositivos y periféricos al robot, en este trabajo se conecta la torreta con la cámara de video y audio inalámbrica, se explica en el apartado 3.5.

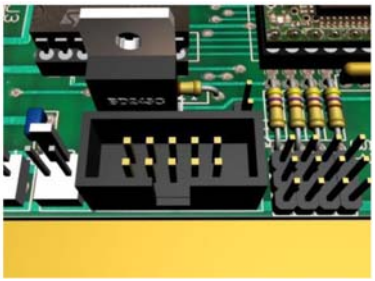


Fig. 3.33 *Conector de expansión*

3.4 Montaje del chasis

En el montaje del chasis se ha procedido a montar las ruedas, servomotores, circuitos y como no el chasis del propio robot, que es de PVC amarillo.

En el manual de instrucciones[1] viene detalladamente cómo se procede a la construcción del chasis. Aunque como se ve en el apartado 6, el manual no está tan detallado como en un principio se puede imaginar y hace omisión a material necesario para la construcción.

Para conseguir que las dos ruedas de tracción puedan girar 360°, en vez de 270° que es hasta donde pueden recorrer los servomotores, se tiene que modificar dos servomotores. Se tiene que abrir los servomotores y cortar el tope que tiene el engranaje principal. De esta manera se consigue que el servomotor pase de girar 270° a poder girar completamente.

En la figura 3.34 se muestra el chasis del robot una vez montado y echa la construcción de él.

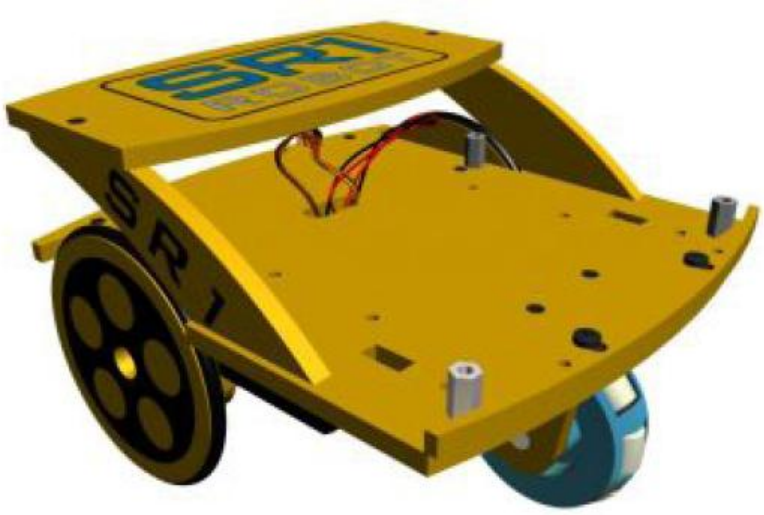


Fig. 3.34 *Chasis del robot*

En la construcción del chasis se encuentran ciertos problemas y para solucionarlos se tiene que rebajar ciertas piezas. En la parte donde están situados los servomotores, se tiene que rebajar ya que el chasis roza con las ruedas.

La pieza que hace de aguante de los ultrasonidos, en la placa, tiene que rebajarse prácticamente a la mitad ya que sino, no entra.

3.5 Posibles ampliaciones del robot

En la figura 3.35 se muestra como queda el robot una vez finalizado todo el montaje.

Fig. 3.35 *Robot al completo sin ampliación*

Con esta configuración, el robot solo sirve para reconocer el terreno, evitando los obstáculos. Por eso se decide ir un paso más allá y en este proyecto, se decide colocarle una cámara. Para ello se tiene que modificar considerablemente el robot. Ante todo se tiene que hacer la construcción de una torreta donde situar la cámara, el sensor de ultrasonidos y 4 leds de alta potencia.

3.5.1 Ampliación de la cámara

3.5.1.1 La cámara

En la cámara se tienen que cortar los cables y se opta por conectarla directamente a la batería del robot. La cámara está situada en una torreta donde se podrá mover, gracias a los servos que se añaden, en las posiciones de vertical y horizontal. Para poder filmar en la posición que se desee, tanto horizontal como vertical.

3.5.1.2 Los ultrasonidos

El sensor de ultrasonidos es colocado en el centro de la torreta de la cámara como muestra la figura 3.36. De esta manera se consigue realizar barridos, tanto en vertical como en horizontal, para que el robot pueda detectar posibles obstáculos que pueda encontrar en su camino.

3.5.1.3 Leds de alta potencia

Los leds de alta potencia situados en la torreta de la cámara, se utilizan para realizar señales llamativas, para que se vean a plena luz del día y a una distancia considerable. Cuando hay escasez de luz, sirven para hacer de foco para el robot e iluminar su camino.

En la figura 3.5.2 se muestra como queda la construcción de la torreta, donde están situados la cámara, los ultrasonidos y los cuatro leds de alta potencia.

Fig. 3.36 *Ampliación de la torreta de la cámara*

En la figura 3.5.3 se observa el resultado final al unir la torreta de la cámara con el chasis del robot.

Fig. 3.37 *Robot completo*

3.5.2 Ampliación juego de orugas

En este kit cabe la posibilidad de ampliar el robot mediante el juego de orugas. Con este juego el robot varia de velocidad, va más lento, a causa de que el diámetro de las ruedas es menor que las ruedas principales de tracción. Cuando se le añaden este tipo de ruedas, lo que se consigue es que el robot tenga un mejor agarre sobre el suelo y sobretodo tenga una mejor reacción sobre el terreno.

En la figura 3.5.4 se puede observar el robot con el juego de orugas.

Fig. 3.38 *Robot con juego de orugas*

3.5 Calibración del robot

Una vez terminada la parte del montaje del robot, el siguiente paso es calibrar los distintos sensores, dispositivos y servomotores del robot SR1. Para ello el fabricante pone a disposición del usuario diferentes programas en BasicX para agilizar estos pasos.

3.5.1 Calibración de los servomotores del robot:

En primer lugar hay que centrarse en los servomotores 1 y 2, que son los que hacen que el robot se mueva. Es necesario calibrarlos para que ambos compartan la misma posición de reposo, ya que si no se hace este paso, cuando el robot le da una orden de movimiento al servo, este no la efectúa correctamente. El motivo de este error viene dado por que, al haber sido modificados en el proceso de montaje del robot, los servos giran 360º y no 270º como lo hacía antes de la modificación, consiguiendo que las ruedas giren sin parar y no tengan un punto de reposo definido.

Para calibrar los servos se dispone del programa **"Servoset"**, el cual permite encontrar el valor del punto de reposo de los servos 1 y 2, y guardarlos en las posiciones de memoria 101 y 102 de la Eeprom del procesador respectivamente. De esta manera, siempre que se comience cualquier programa para el robot, se podrá leer la posición de la Eeprom al principio de los programas y obtener los valores de parada siempre que sean necesarios.

Una vez se ha descargado el programa en el Basicx24, se puede apreciar en la pantalla del entorno de programación, que aparece un valor asignado al Servo1, y que a su vez, la rueda izquierda comienza a girar hacia atrás. En ese momento, y con el robot en la mano, se debe comenzar a variar el valor asignado al Servo1 pulsando el parachoques izquierdo del robot, el cual incrementa en una unidad el valor asignado al servo hasta que éste reduzca su velocidad. En esta parte se ha de tener cuidado y mucha precisión, ya que se ha de dejar de apretar el parachoques cuando el número que aparece por pantalla sea 1400 exactamente.

Una vez se llega a dicha posición, se pulsa una única vez el parachoques derecho para comenzar a calibrar el servo2, de la misma manera que se hizo con el anterior servo. Cuando se terminan de calibrar los dos servomotores, los puntos de reposo son mostrados en la pantalla del entorno de programación y guardados en las posiciones de la Eeprom anteriormente mencionadas.

Como se puede comprobar, los servos no están en su posición de reposo, moviendo las ruedas del robot hacia adelante o atrás. En ese momento se han de desmontar los dos servos y, mediante un destornillador pequeño, se debe de variar con mucho cuidado el potenciómetro de los servos hasta que estos dejen de moverse, lo que indica que están situados en su posición de reposo.

3.5.2 Calibración de los servos de la cámara:

Una vez se han calibrado correctamente los servos del robot, hay que calibrar los Servos 3 y 4, que son los encargados de mover la cámara del robot de forma vertical y horizontal respectivamente. Para hacer esta calibración se debe usar el programa **"Camset"**.
Cuando se ha terminado de cargar el programa en el Basicx24 del SR1 y se ejecuta, se comienza a ajustar el servo que se encarga de los movimientos verticales.
La mecánica de este programa es idéntica a la usada con el programa **"Servoset"**, por lo que la calibración del servo vertical se debe comenzar pulsando el parachoques izquierdo del robot hasta que la cámara comienza a levantarse, en ese instante, se deja de pulsar el parachoques izquierdo y se pulsa el parachoques derecho para que el programa guarde la posición en la Eeprom. Acto seguido, se pulsa el parachoques izquierdo de nuevo hasta que la cámara llega a su punto máximo de elevación, que es el momento en el que la cámara no sube más por mucho que se pulse el parachoques. En ese momento, se vuelve a pulsar el parachoques derecho para guardar la posición en la memoria y finalizar la calibración del servo vertical, pasando en este momento a la calibración del servo horizontal.
Dicho servo se ha de ajustar a la posición central de la cámara, y 90º hacia la izquierda y derecha desde el punto central calculado anteriormente.
Para ajustar el punto central, se debe de pulsar el parachoques izquierdo hasta que la cámara se encuentra perfectamente centrada con respecto a la base del robot. Luego, se pulsa el parachoques derecho y se ajusta la posición de 90 grados a la izquierda y después derecha, teniendo como referencia la posición central.
Una vez se han terminado de hacer estos pasos, la calibración de los servos de la cámara está finalizada, y como en el apartado anterior, los puntos calibrados son guardados en la memoria Eeprom del Basicx24 a la espera de ser usados por algún programa.

3.5.2.1 Modificación del programa para calibrar los servos de la cámara

En la calibración anterior hay varios problemas con el servo encargado de subir y bajar la cámara, el servo 3. Al comenzar la calibración, el servo 3 funciona de forma errónea cuando se intenta calibrar, por lo que después de comprobar su correcta colocación se ha de modificar el programa de calibración.
El fallo consiste en que, cuando el servo debe de subir la cámara el programa de calibración le ordena que la baje, por lo que la cámara no se eleva. Igual resultado se obtiene cuando se intenta calibrar el punto mínimo de la cámara, ya que el programa de calibración le ordena al servo que baje la cámara.
El error y la solución del código se pueden apreciar en la figura 3.39.

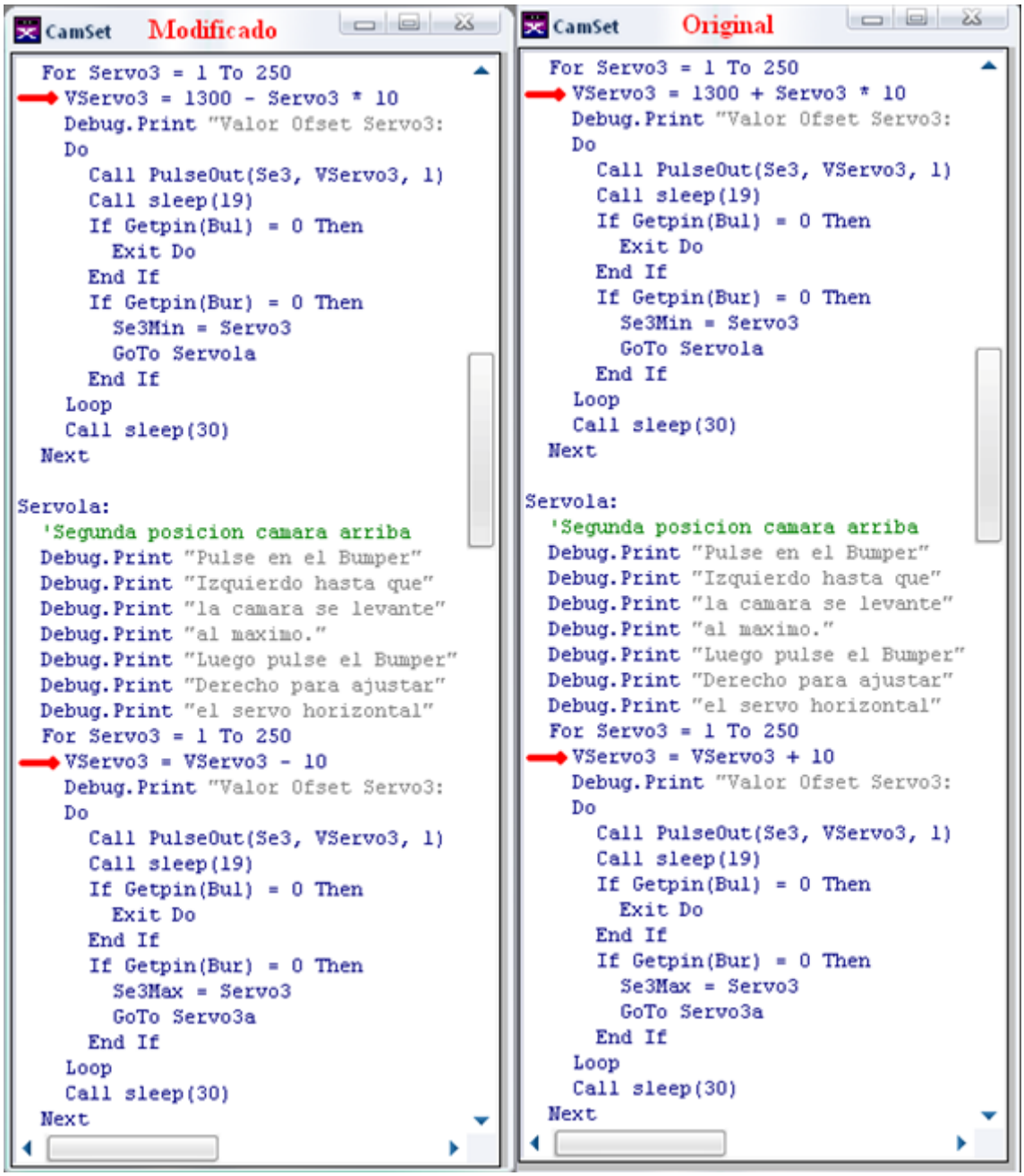


Fig. 3.39 *Modificación del programa de calibración de la cámara*

En la imagen, se puede observar el programa de la izquierda ya modificado y la de la derecha sin modificar, pudiendo apreciar dos marcas en cada programa, las cuales señalan el error.
El problema radica en que el programa sitúa el servo en una posición incorrecta para que suba y baje la cámara, teniendo los dos símbolos de la operación puestos de manera incorrecta. Al cambiar los símbolos de suma por los de resta se obtiene la correcta calibración.

3.5.3 Testeo de los Leds:

El SR1, a parte de los leds de infrarrojos, también dispone de otros dos leds de iluminación para los momentos de baja luminosidad. Para comprobar el correcto funcionamiento de estos, el manual propone un programa sencillo para testearlos, el cual se muestra en la figura 3.40.

```
Option Explicit
Const Led as Byte = 9       'Pin del Led
Const Bur as Byte = 18      'Pin Pulsador Izquierdo
Const Bul as Byte = 19      'Pin Pulsador Derecho
Public Sub Main()           'Rutina principal
Do                          'Bucle
  If GetPin(Bul) = 0 then   'Si Izquierdo pulsado
    Call PutPin(Led, 0)     'Apaga los leds
  End If
  If GetPin(Bur) = 0 then   'Si Derecho pulsado
    Call PutPin(Led, 1)     'Enciende los leds
  End If
Loop                        'Repite el bucle
End Sub
```

Fig.3.40 *Programa para testear los leds*

Este programa, una vez se ha volcado en el Basicx24, da la orden de encender los leds al pulsar el parachoques izquierdo, y desconecta los leds al pulsar el parachoques derecho. Si el programa funciona de manera satisfactoria significa que los leds funcionan de manera correcta.

3.5.4 Testeo de los infrarrojos:

En este apartado se comprueba el funcionamiento correcto de los infrarrojos. En primer lugar, para que estos funcionen, se deben de puentear los pines 10, 11 y 12 de los conectores J6 y J8. Acto seguido, se vuelca en el Basicx24 el programa **"Infrarrojos"**, el cual muestra en la pantalla del entorno de programación la distancia a la que está situado un objeto de los dos leds infrarrojos, encendiendo a su vez el led verde del Basicx24 si hay un objeto cerca del led izquierdo y el rojo si lo está del derecho. Si está cerca de ambos leds infrarrojos se encienden los leds rojo y verde a la vez.
La mejor forma de comprobar que están funcionando correctamente los leds es colocar la mano cerca de los leds e ir alejándola y acercándola a estos, viendo como en la pantalla del entorno de programación la distancia va aumentando y disminuyendo.
Para terminar y como nota curiosa, el usuario no puede saber si los leds infrarrojos están en funcionamiento o no, ya que el ojo humano no es capaz de ver la luz infrarroja. Una buena manera de averiguarlo es grabar el robot con una filmadora o hacerle una fotografía, ya que en ella sí se ven encendidos.

3.5.5 Testeo de las células fotoeléctricas:

El robot dispone de dos células fotoeléctricas situadas en las esquinas delanteras del chasis, para comprobar su correcto funcionamiento se usa el programa **"LecturaLDR"** ,el cual una vez está volcado en el robot SR1, muestra por pantalla los niveles de luz que toman las dos células fotoeléctricas. Para comprobar el correcto funcionamiento de los sensores se deben tapar cada una de las células, observando que cuando están tapadas el valor numérico que se muestra en la pantalla aumenta significativamente.

3.5.6 Testeo del sensor de ultrasonidos y del sensor de luminosidad:

Para comprobar el correcto funcionamiento del sensor de luminosidad y ultrasonidos se dispone del programa **"Ultrasonido"**, el cual funciona de una forma similar al programa para testear los leds de infrarrojos.

Una vez se ha volcado el programa en el Basicx24 se pueden apreciar dos valores en la pantalla del entorno de programación, el primero corresponde al sensor de ultrasonidos, el cual puede comprobarse su funcionamiento de la misma manera que los leds infrarrojos, ya que si alejamos y acercamos nuestra mano o un objeto al sensor, veremos cómo los valores numéricos van aumentando o disminuyendo respectivamente.

El segundo dato que aparece en la pantalla del entorno de programación es el del sensor de luminosidad, del cual se hace el testeo de la misma forma que con las células fotoeléctricas, observando cómo al tapar el sensor de luminosidad el dato numérico que aparece en la pantalla del entorno de programación aumenta, y al destaparlo disminuye. Si ambas pruebas han resultado satisfactorias quiere decir que los dos sensores funcionan correctamente.

3.5.7 Calibración y testeo de la brújula digital

La última de las calibraciones es la de la brújula digital. Para realizarla se utiliza el programa llamado **"Brújula"**, el cual muestra la dirección de la brújula en la pantalla del PC mediante un valor numérico en grados, y ayudara a calibrarla mediante los parachoques.

Este programa obliga al usuario a tener el robot en una posición perfectamente horizontal y que este alejado de objetos metálicos y magnéticos para que la calibración sea correcta.

En primer lugar se debe disponer de una brújula convencional, la cual es usada para hacer una calibración precisa de nuestro sensor. Se debe comenzar por situar el robot con precisión hacia el norte. Una vez comprobado con la brújula convencional que está correctamente colocado se debe pulsar el parachoques izquierdo para que el robot guarde el dato en la memoria Eeprom de donde está situado el norte. A continuación se coloca el robot hacia el Este, realizando los mismos pasos que cuando se calibra la posición Norte, al finalizar se hace lo propio con el Oeste y con el Sur.

Una vez terminado el proceso, en el mismo programa se puede ver como al girar el robot los grados que se muestran en la pantalla del entorno de programación van variando. Una buena manera de comprobar que el programa funciona es volcar en el robot el programa **"Explorer"** y usar la opción de **"girar al norte"**. Si después de girar el robot hacia dicha posición esta coincide con el valor que nos da la brújula convencional, quiere decir que la calibración ha sido optima.

4 ENTORNO DE PROGRAMACION Y PRESTACIONES

El robot SR1 se programa mediante el entorno de programación BasicX, el cual permite escribir, compilar y descargar los programas para ser usados por el microcontrolador BasicX24.
La programación del robot es similar al de cualquier sistema basado en un procesador externo, creando un texto que contenga el código del programa y a continuación se emplea un compilador que genera el código hexadecimal y otros ficheros con datos adicionales.
El entorno de programación, facilita este proceso ya que cuenta con una serie de herramientas y opciones que permiten seleccionar los parámetros de trabajo fácilmente.
Una de las ventajas que tiene este tipo de entorno de programación, el cual se aprecia en la figura 4.1, es que tiene muchas similitudes con Visual Basic, ya que ambos entornos de programación comparten las mismas instrucciones, aunque con unas cuantas diferencias.

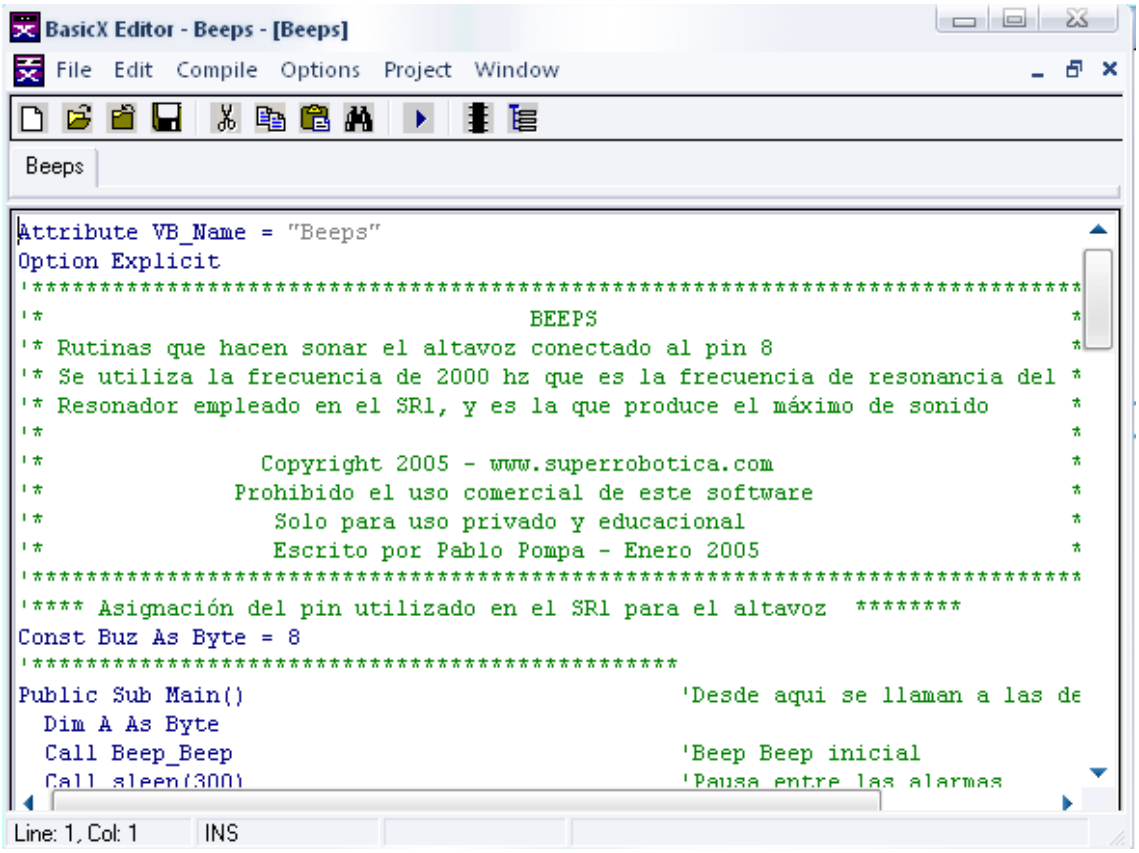


Fig.4.1 *El entorno de programación.*

Por ejemplo, en BasicX no existen los formularios, el programa tiene que ser compilado y descargado sobre el robot antes de que pueda ejecutarse, lo que alarga sensiblemente el proceso de depuración. Se puede ver con detenimiento en la figura 4.2 el entorno BasicX para descargar los programas creados en el BasicX24.

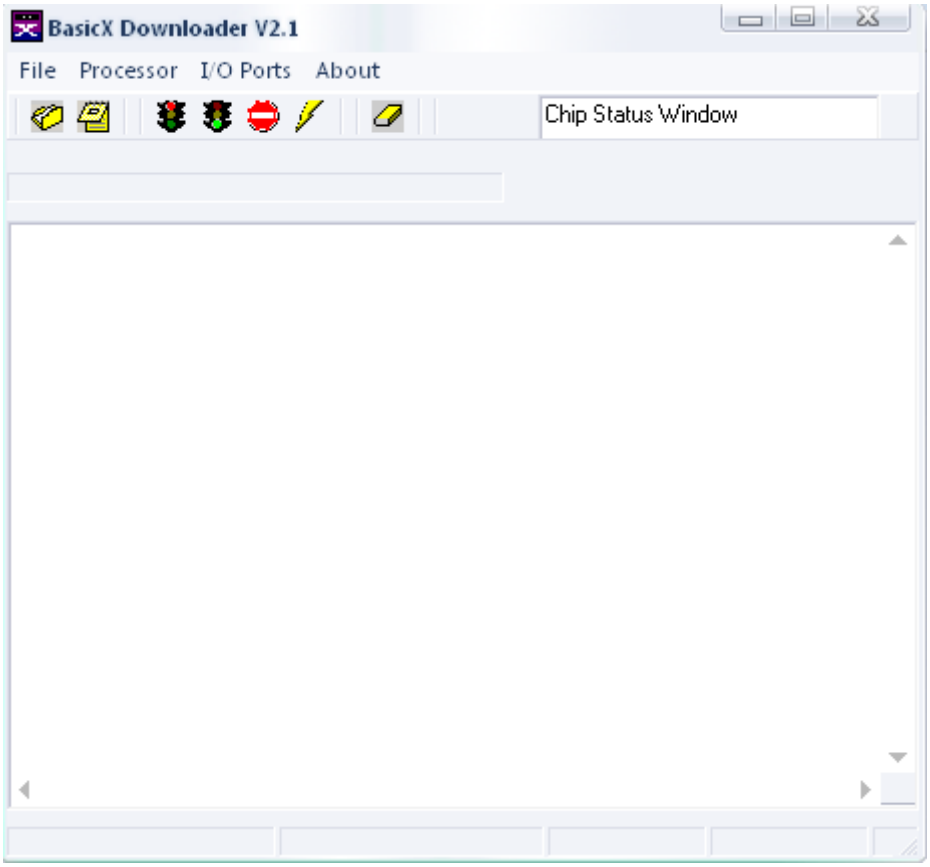


Fig.4.2 *Aplicación para transferir el programa al robot.*

Aunque la mayor diferencia entre ambos radica en el procesador, siendo el BasicX24 un potente microcontrolador de 8 bytes que cuenta con la ayuda de un puerto de comunicaciones, 3 temporizadores, 32 K de memoria y poco mas, este no tiene pantalla, ni teclado, ni discos, etc. Además el procesador se encuentra fuera del entorno de programación, en otro dispositivo, por lo que hay que mandarle el programa primero por el puerto serie para que lo pueda ejecutar.

En cambio, cualquier PC moderno cuenta con un procesador de 32 bytes que es mil veces más rápido y un millón de veces más potente, además está acompañado por coprocesadores que se encargan de controlar toda clase de periféricos como pantallas, discos, teclados, puertos, etc. Igualmente el procesador destinatario del programa es el mismo que lo está realizando, y por lo tanto se puede hacer una ejecución inmediata y tener acceso a todas las variables instantáneamente para su depuración.

Aunque el entorno de programación de BasicX no tiene esta característica, el hecho de que este entorno de programación sea compatible con Visual Basic permite abrir el fichero con extensión **'bas'**, el cual contiene el código fuente del programa, desde Visual Basic para su edición. Además se corregirá automáticamente la escritura de todas las variables para que queden escritas de forma homogénea a lo largo del programa. Una vez corregido y terminado, se guarda de nuevo con la opción Control +S o bien con la opción 'guardar archivo.bas' dentro del menú 'Archivo de Visual Basic'.

Hay varias recomendaciones a la hora de programa con BasicX, como por ejemplo comentar los programas que creamos para una futura comprensión del trabajo por parte propia o por parte de otro programador que quiera modificar o comprender el programa base, característica usada en prácticamente todos los entornos de programación y que en BasicX realizamos con el comando **'** antes de hacer los comentarios, ya que si no se mezcla el comentario con el código escrito y hace que el programa no funcione o lo haga de forma defectuosa.

Otra recomendación importante es la de no utilizar directamente los números de las patillas del procesador BasicX24, estas se deben asignar a constantes que son las que hay que utilizar en el código. Esto no solo hace mucho más fácil y entendible el código, si no que permite cambiar fácilmente los pines utilizados en cada función por otros, con solo cambiar el valor que se le asigna a la constante al principio del programa, sin tener que actualizar todo el código en sí.

Las constantes, a diferencia de las variables, se sustituyen por sus respectivos valores durante la compilación, lo que significa que no consumen recursos ni memoria del procesador. Las variables por el contrario, están limitadas por la cantidad de memoria que tiene el procesador, por lo que hay que tenerlo en cuenta cuando se realizan los programas.

4.1 Programas específicos para los sensores

En este apartado se comentan los programas que nos facilita el fabricante para diversas aplicaciones del SR1. Dichos programas son usados para que el usuario comprenda el funcionamiento de los sensores con un ejemplo práctico, y para que, al abrir el programa con el editor, se puedan entender la forma de inicializarlos en el programa y hacerlos funcionar de una forma óptima.

4.1.1 Programa NavBasica1

Es el programa más sencillo de todos ellos y el primero que se debe usar para comprobar que se han calibrado correctamente los servomotores y que los parachoques funcionan correctamente.

La función principal del programa, es hacer que el robot avance en línea recta hasta que los parachoques se topen con un obstáculo, y dependiendo de si se ha activado el parachoques izquierdo o derecho, efectúe un giro hacia la izquierda o hacia la derecha. La principal desventaja de este programa es cuando el robot se encuentra con una esquina, ya que cuando el robot colisiona con el parachoques izquierdo y gira hacia la izquierda, el parachoques derecho colisionara con el otro extremo de la esquina haciéndole girar hacia la derecha y haciendo que el robot este constantemente colisionando, siendo este incapaz de salir de la situación hasta que se le terminan las pilas. Como es un programa más de testeo y para familiarizarnos con las funciones del robot, este no es un problema real importante, ya que, como ve en programas posteriores, este tipo de problemas son solucionados por el robot mejorando el código del programa.

El código de este programa es muy simple, ya que lo único que hace es inicializar las variables y los pines necesarios, y hacer un programa principal sencillo, el cual comienza recuperando los valores de parada de los servos, que a continuación aparecerán en la pantalla del PC gracias a la conexión del robot por medio del puerto serie una vez se vuelca el programa en este y coloca los servos en dicha posición. El siguiente paso es realizar un bucle infinito, el cual simplemente se encargara de ver si el parachoques izquierdo esta pulsado para llamar a la subrutina **“GiroI”** (gira el robot a la izquierda) o si esta pulsado el derecho para llamar a **“GiroD”** (Gira el robot a la derecha). Si no están pulsados ninguno de los dos parachoques el robot avanza en línea recta.

4.1.1.1 Valoración de NavBasica1

Este programa es una buena toma de contacto con el robot, aunque el programa es muy simple, sirve para ver si el robot avanza en línea recta sin desviarse y para comprobar que al chocar con un obstáculo el robot gira. El testeo del programa es optimo y funciona perfectamente, salvo por el defecto del “efecto esquina” comentado en el apartado anterior.

4.1.2 Programa NavBasica2

Este programa no es más que una mejora del programa anterior, pero añadiéndole el uso del altavoz del robot por primera vez, haciendo que cada vez que se conecte el robot y en este esté volcado el programa, el robot efectúe dos pitidos. Estos los efectúa cuando el sensor de inclinación detecta que el robot esta en desnivel, usándose también en el programa el sensor de inclinación, el cual sirve para detectar obstáculos que sean más pequeños que la altura de los parachoques y que, por lo tanto, nunca llegan a colisionar con estos, haciendo que el robot quizás no pueda pasar por encima suyo, quedando bloqueado hasta que se terminan las baterías.

Gracias al sensor de inclinación, este problema puede ser solucionado, ya que si el sensor detecta que el robot se encuentra en desnivel, llama a una nueva subrutina llamada **"Atrás"** (la encargada de que los servos muevan marcha tras el robot) para que este retroceda hasta bajarse del obstáculo, y posteriormente, llama a la subrutina **"Giro D"** para que sortee el obstáculo. Otra de las mejoras respecto al programa anterior es que el programa comprueba si los dos parachoques están pulsados a la vez, lo que hace que el robot llame a la subrutina **"Atrás"** y posteriormente a la subrutina **"GiroD"** para girar el robot hacia la derecha.

La última mejora es para solucionar el problema del "efecto esquina" mencionado en el programa anterior, ya que esta vez los servos contaran con un contador que se va incrementando continuamente por el robot mientras este avanza en línea recta. Dicho contador es el que le dice al programa si cuando el robot colisiona con un obstáculo este estaba avanzando en línea recta o, por el contrario, ya había colisionado con el otro parachoques y estaba siendo afectado por el "efecto esquina". Este problema se soluciona haciendo una modificación en la subrutina **"GiroI"**, que es la encargada de hacer girar al robot hacia la izquierda, y que con la nueva modificación, comprueba antes de efectuar el giro si el robot había girado la vez anterior a la derecha, y en caso afirmativo, efectuara un giro completo de 180º para salir de la esquina.

4.1.2.1 Valoración de NavBasica2

Con el programa **NavBasica2** tampoco hay ningún fallo a la hora de probarlo en el robot, el único defecto viene dado en la parte de hardware, ya que el robot está continuamente chocándose con objetos y eso, a la larga, modifica la posición del sensor de inclinación, lo que ocasiona que el robot crea que está en desnivel y esté continuamente esquivando un obstáculo inexistente. Las mejoras que trae este programa son notables, ya que evita el efecto esquina perfectamente y el sensor de inclinación funciona a la perfección a la hora de comprobar que el robot está en desnivel.

4.1.3 Programa NavIR

El programa es una mejora de **"NavBasica2"**, en este programa se comienza a hacer uso de la detección de obstáculos a distancia a través del sensor de infrarrojos. Para que dicho programa funcione, lo primero que se hace es puentear los conectores J6 y J8 de los pines 10, 11 y 12 del robot. El pin 11 conecta el infrarrojo izquierdo, el 11 el derecho y el 12 el receptor IR. Las modificaciones del programa principal comienzan en el bucle infinito, el cual, lo primero que hace, es realizar una lectura de los dos leds infrarrojos mediante las subrutinas **"LeeIR1"**, dicha subrutina comprueba si el led 1 ha leído algo, y **"LeeIR2"** hace lo propio con el led 2. Es importante detallar lo que realizan las subrutinas para legar a comprender cómo funciona el programa, por lo que son explicadas con detenimiento y de forma general, ya que la única diferencia entre ambas, es el cambio de los parámetros mandados a la instrucción para encender, leer y apagar los leds.

La subrutinas comienzan encendiendo el led, acto seguido ponen a cero la variable que le se le pasa como parámetro al programa principal, la cual si es menor de 5 significa que hay un obstáculo, y si es mayor significa que el camino está libre. Una vez puesta la variable a 0 el programa entra en un bucle que realiza 6 mediciones del led, si el led no encuentra ningún obstáculo, el valor que adquiere la variable es de 6, en cambio, con que en una sola medida localice algún objeto, la variable pasará a ser menor a dicho número.

Una vez el programa principal recibe la respuesta de la variable para leer el led en forma de variable entera, este pasa a analizarla, evaluando la condición de que si la variable es mayor de 6 el robot sigue en línea recta y si es menor, enciende el led verde o rojo del microprocesador, dependiendo del led infrarrojo que encuentra el obstáculo, y efectúa un giro hacia la izquierda o derecha dependiendo también de los infrarrojos. Por último, termina por apagar los leds del microprocesador para, una vez más, volver a leer los datos de los leds y comienza de nuevo con el bucle infinito.

4.1.3.1 Valoración de NavIR

Este programa ha sido testeado también de forma correcta, pero con algún que otro error en las primeras pruebas, ya que si las baterías del robot están un poco gastadas o si el termo retráctil obstaculiza un poco los leds de infrarrojos el robot funciona de manera incorrecta. En este programa se han efectuado diferentes pruebas editándolo para variar la distancia que miden los infrarrojos, para de esta forma probar el alcance de estos y su precisión con objetos lejanos. El robot responde bien cuando el objeto está a medio metro aproximadamente y evita el obstáculo perfectamente, pero en espacios reducidos pierde mucha efectividad.

4.1.4 Programa Navegación buscando Luz

Este programa tiene un esqueleto parecido al programa **"NavIR"**, solo que en este caso el sensor que se usa son las dos fotorresistencias que hay en los extremos frontales del robot SR1. Lo primero que se ha de hacer para que el robot use las fotorresistencias, es puentear los pines 11 y 12 de los conectores J7 y J8, de esta manera se consigue activar las fotorresistencias para usarlas en el programa.

El programa básicamente mide la luz mediante la llamada a dos subrutinas que leen las fotorresistencias, para calcular la diferencia entre las medidas de los dos extremos y gira el robot hacia el lado donde hay más luz. Este programa también está explicado paso a paso, para que no se quede con una idea general de él y se entienda perfectamente.

El programa principal comienza recuperando los valores de parada de los servos y colocándolos en dicha posición, acto seguido el SR1 emite dos pitidos para indicar que este va a empezar con el bucle infinito principal del programa. Dicho bucle comienza haciendo una llamada a la primera fotorresistencia y obtiene su valor. En ese momento, los guarda en una variable mediante la instrucción **RCTime (Pin , Estado)**. Lo siguiente que realiza el programa es hacer la misma operación anterior para guardar los datos de luminosidad de la fotorresistencia 2 en otra variable.

Una vez obtiene los dos valores de luminosidad pasa a comprobar que fotorresistencia ha encontrado una mayor cantidad de luz. Para ello evalúa la resta de las dos variables, para ver si es más grande que el numero entero 100, valor que se ha escogido para darle un margen de error a la resta, ya que, como es normal, los valores captados por las dos fotorresistencias, aunque el usuario no emita luz, son distintos y harían que el robot estuviera girando siempre.

Si la resta de las variables es mayor que 100 este hace que el robot gire hacia la izquierda, y si la resta de las variables 2 y 1 es mayor a 100 gira hacia la derecha, si ninguna de las dos restas es más grande de 100 significa que no hay la suficiente luz como para que el robot gire en una dirección, por lo que el robot comienza a avanzar hacia delante en busca de luz. El resto del programa es como **"NavBasica2"**, donde se comprueba el estado de los parachoques evitando el "efecto esquina" y la inclinación del robot mediante el inclinómetro.

4.1.4.1 Valoración de Navegación buscando Luz

Un problema que tiene este programa, es el numero que evalúa la condición de la variable, ya que en ocasiones el robot gira sin enfocarlo con luz, bien por qué las baterías están bajas y los valores recibidos comienzan a ser poco fiables, o bien por el ángulo en que están puestas las fotorresistencias, ya que si el ángulo es muy cerrado y las dos están próximas, enfocar con una linterna de amplias dimensiones es un problema, ya que abarca las dos fotorresistencias. Cuando se soluciona el problema y se usa una linterna de pequeñas dimensiones el testeo del programa es optimo y funciona correctamente.

4.1.5 El programa NavUltrasonidos

El programa **"NavUltrasonidos"** tiene la misma finalidad que el programa **"NavIR"**, solo que esta vez, en vez de usar los sensores de infrarrojos para detectar obstáculos a distancia, se usa el sensor de ultrasonidos. Cada tipo de sensor funciona mejor dependiendo de la superficie del objeto a detectar y de la distancia a la que está el objeto, en este caso, los resultados más satisfactorios se logran con los ultrasonidos, obteniendo menos errores en la detección de obstáculos.

El programa principal comienza situando los servos en la posición de parada, y declarando una variable entera de valor 30 con el nombre de **Rango**, dicha variable es la que marca la distancia en centímetros a la que el programa detecta un obstáculo. Acto seguido inicializa los leds del SR1 apagándolos y hace sonar la alarma antes de entrar en el bucle principal.

Una vez el programa entra en el bucle principal infinito, su primera acción es llamar a la subrutina **"Distancia"**, que es la encargada de leer los datos de la distancia a la que están los objetos de los ultrasonidos del SR1, y el valor de luminosidad mediante el sensor lumínico que está próximo a este. En ese momento, evalúa la condición de la luminosidad viendo si el dato leído por distancia es menor al número entero 60, si lo es, quiere decir que el nivel de luz es bajo y enciende los leds del SR1, y si es mayor, los deja apagados.

Acto seguido el programa evalúa el dato que le ha pasado el sensor de ultrasonidos mediante la subrutina **"distancia"**, si este es menor al valor de la variable **Rango** inicializada anteriormente a 30 cm, indica que hay un obstáculo en frente del robot. En ese momento se enciende el led rojo del microprocesador y se incrementa una variable llamada **veces** en una unidad. Dicha variable sirve para que el robot efectúe giros a izquierda, si esta es menor a 2, o a la derecha, si es mayor.

A continuación, comprueba que la variable nunca sobrepase el valor entero 4 reseteándola, ya que si no, una vez supere el valor entero 2, el robot siempre gira hacia la derecha cuando encuentra un obstáculo. El propósito de esta variable es que, a diferencia del programa con infrarrojos, en este no se puede saber en qué lado se encuentra el obstáculo, ya que solo se dispone de un sensor de ultrasonidos y este está en la posición central del robot.

En la parte final del programa, se siguen haciendo mediciones de la distancia hasta que el robot esquive el objeto, para confirmar que el giro ha sido suficiente, una vez la distancia es mayor de 30 cm sale del bucle y apaga el led rojo del microprocesador.

A continuación el programa continua comprobando los parachoques y el sensor de inclinación como en programas anteriores, y si no ha encontrado ningún objeto cercano o ha colisionado, avanza en línea recta.

4.1.5.1 Valoración de NavUltrasonidos

Este programa funciona correctamente y los resultados con el son asombrosos, obteniendo muy pocos fallos en colisiones. Gran parte del merito del éxito del programa lo tiene el sensor de ultrasonidos, el cual detecta los obstáculos de una forma mucho más precisa que los infrarrojos. El único problema que tiene esta aplicación es también en la parte de montaje, ya que si el robot dispone de la expansión de la cámara, hay que quitar la pieza de sujeción de los ultrasonidos de la placa, ya que si no, los sensores la detectan como un obstáculo y el robot no deja de girar continuamente.

4.1.6 El programa EXPLORER

4.1.6.1 Explicación general

El programa EXPLORER se basa en controlar el robot mediante un programa a nivel de usuario, como se puede ver en la figura 4.3. Con este programa lo que se consigue es que el usuario pueda dominar el robot, que está conectado mediante un cable de puerto serie con el ordenador, enviando las órdenes a través del cable y de esta manera el robot ejecute los comandos pedidos por el usuario a través del ordenador.

El programa Explorer está compuesto por dos tipos de software, el primero sirve para programar el microprocesador, que está en el robot, hace que el robot reciba la información mediante el puerto serie y acto seguido haga el movimiento deseado por el usuario. El otro programa es para que el usuario pueda ejecutarlo en el PC, y actúa como base del control y envía los comandos por el puerto serie para que así el robot pueda ejecutar la acción deseada.

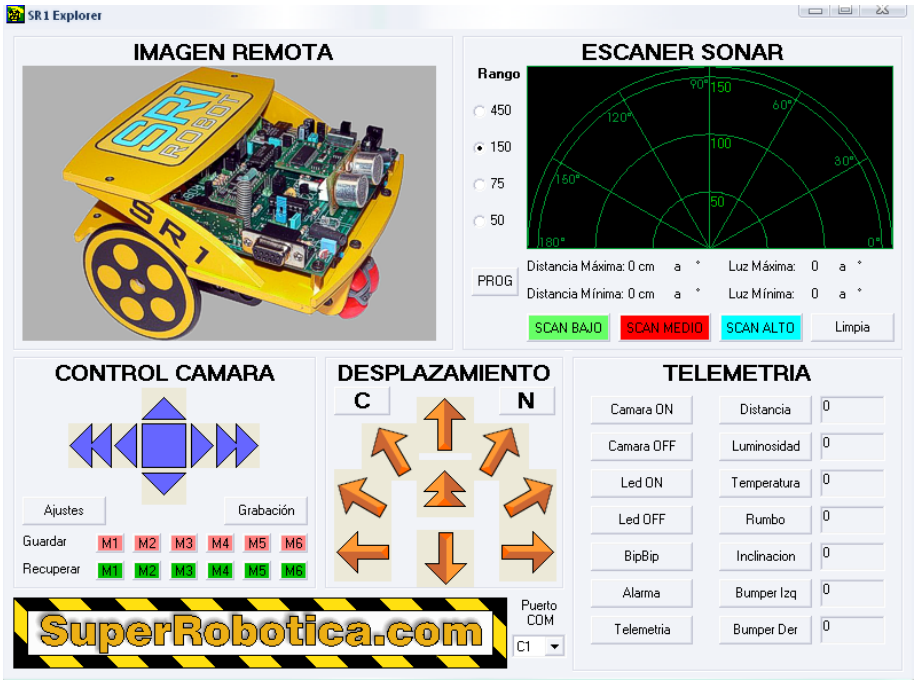


Fig. 4.3 *Programa de control SR1Explorer del robot SR1 mediante el ordenador*

4.1.6.2 Explicación del programa principal

El programa que se ejecuta en el robot, dispone del programa principal. Lo primero que hace es preparar los búfers de recepción y transmisión del puerto serie y el puerto Com1, para que el robot pueda recibir los comandos que el usuario le está mandando a través del ordenador.

Se asegura de que tanto los servos de la cámara como los del robot estén en su posición central y parada, que la cámara esté desactivada y los leds estén apagados. Cuando todo ya está prácticamente listo hace un beep para decir que ya está preparado.

Una vez ya está todo preparado, se espera a recibir un número de caracteres por el búfer de recepción del puerto serie.

Cuando los recibe, comprueba el número de caracteres recibidos por el búfer de recepción del puerto serie. Acto seguido hace una comparación y comprueba que si el número de bytes recibidos en el puerto serie es mayor a cuatro, entonces hay un comando. Si se da el caso, lee cinco caracteres del puerto serie y asigna a la variable comando los cinco caracteres leídos. Seguidamente asigna a la **subrutina SubComando** los tres primeros caracteres recibidos en el puerto serie para identificar la familia del comando.

4.1.6.3 Explicación de las subrutinas

El programa Explorer está compuesto por varias subrutinas, que se agrupan en cuatro apartados:

- **4.1.6.3.1 La subrutina SubComando** en la que se detallan las funciones con las que se dota al robot.

- **4.1.6.3.2 Las subrutinas de movimientos de la cámara** son las que detallan los movimientos y acciones que puede realizar la cámara.

- **4.1.6.3.3 Las subrutinas de movimientos del robot** son las que detallan los movimientos y acciones que puede realizar el robot.

- **4.1.6.3.4 Las subrutinas para que el robot realice la telemetría** son con las que se detallan los resultados obtenidos mediante los sensores que forman el robot.

4.1.6.3.1 La subrutina SubComando

En la subrutina **SubComando** están declarados los comandos de movimiento del robot con los caracteres **MOR.** Los comandos de movimiento de la cámara con los caracteres **MOC.** Los comandos de memorizar la posición de la cámara con los caracteres **REC.** Los comandos de recuperar la posición de la cámara con los caracteres **CAL**. Los comandos de telemetría con los caracteres **TEL**. Los comandos de recibir programa con los caracteres **TXP**. Los comandos de ejecutar programa con los caracteres **EJP**.

Todas las familias de comandos que pertenecen a la subrutina **SubComando** tienen una instrucción de cinco caracteres asignada a la variable comando, con la que se especifica las acciones que el robot tiene que tener en cuenta.

Con el SubComando (MOR), que son los comandos de movimiento, el robot puede: ir hacia delante **(MORAE)**. Ir hacia adelante más rápido **(MORAA).** Ir marcha atrás **(MORAT)**. Girar a la izquierda o derecha con un pequeño giro **(MORAI, MORAD respectivamente),** con 45 grados **(MORIA, MORDA respectivamente)** o con 90 grados **(MORIZ, MORDE respectivamente).** Girar hacia el norte **(MORNO)** y girar a la posición de la cámara **(MORCA)**.

Con el SubComando (MOC), comandos de movimiento de la cámara, se puede: mover la cámara a la derecha y izquierda **(MOCDE, MOCIZ respectivamente)**. Mover la cámara a la derecha e izquierda más rápido **(MOCDD, MOCII respectivamente)**. Mover la cámara hacia arriba y hacia abajo **(MOCAR, MOCAB respectivamente)**.

Con el SubComando (REC), comandos de memorizar la posición de la cámara, esta subrutina calcula el número de memoria y luego guarda la memoria de la cámara.

Con el SubComando (CAL), comandos de recuperar la posición de la cámara, esta subrutina primero calcula el número de memoria y luego mueve la cámara a la posición central.

Con el SubComando (TEL), comandos de telemetría, puede: comprobar la distancia a la que se encuentra algún objeto por delante suyo y la envía **(TELDI)**. Comprobar la luz que hay en ese momento y envía el valor **(TELLU).** Comprobar la temperatura a la que se encuentra el robot y la enviará **(TELTE)**. Comprobar el rumbo y lo envía **(TELRU)**. Comprobar la inclinación **(TELIN)**. Comprobar el parachoques izquierdo y derecho **(TELIZ, TELDE respectivamente)**.

Con el SubComando (TXP), comandos de recibir el programa, primero calcula el número de programa y luego recibe los datos del programa.

Con el SubComando (EJP), comandos de ejecutar el programa, primero calcula el número de programa y luego ejecuta el programa.

Hay algunos comandos que se le envían al robot que no pertenecen a ninguna familia de **SubComandos**. Con estos comandos se puede encender la cámara con los caracteres **CAMON**. Apagar la cámara con los caracteres **CAMOF**. Encender los leds con los caracteres **LEDON**. Apagar los leds con los caracteres **LEDOF**. Hacer un ruido con el altavoz, beep-beep, con los leds encendidos con los caracteres **ALARM**. Hacer un beep con los caracteres **BIPON**. Hacer un barrido con el sonar en posición baja **SCAN0**, posición media **SCAN1** y posición alta **SCAN2**.

En la subrutina **EJPRogra**, que ejecuta los comandos del programa, se crea un índice de los comandos leídos para así poder saber el que se está leyendo. Crea un bucle en el que primero lee un dato del programa y va repitiendo la secuencia. De esta manera va aumentando el índice y recorrer todos los programas para hacer una comparación con el dato leído. Si la comparación es cierta sale del bucle y llama a la rutina que ejecuta las ordenes. Para evitar reinicios entre instrucciones se hace una pausa.

La subrutina **Procesa**, que ejecuta las órdenes y crea una selección donde se encuentran todos los casos posibles, en los que el robot tiene que obedecer y realizar la instrucción que se selecciona por el usuario previamente con el ordenador.

4.1.6.3.2 Subrutinas de movimientos de la cámara

4.1.6.3.2.1 Subrutinas para el movimiento de la cámara hacia la izquierda o a la derecha

Las subrutina que mueven la cámara a la izquierda o a la derecha se llaman **CIzquierda** o **CDerecha**, en estas subrutinas se crea un bucle que está dominado por una variable entera fija que se encarga de la velocidad a la que se mueve la cámara. Dentro de este bucle es donde existe la diferencia entre la subrutina **CIzquierda** y **CDerecha**, ya que la única operación realizada es la de restar o sumar valores a la posición del Servo4, este servomotor se encarga únicamente de mover la cámara en posición horizontal y de izquierda a derecha. Una vez iniciado el movimiento del servomotor, se hace una llamada a la subrutina CPulsos para ver si supera la posición máxima o mínima del servomotor horizontal, y en caso afirmativo, se vuelve a colocar en un rango posible.

La subrutina que mueve la cámara a la izquierda o derecha rápido, es igual a la subrutina anterior, lo único que varia es la variable que domina al bucle que es el doble de su valor, y por lo tanto, mueve la cámara a la izquierda o derecha con más velocidad.

4.1.6.3.2.2 Subrutina para el movimiento de la cámara hacia el centro

La subrutina que mueve la cámara al centro se llama **CCentro** y se basa en un bucle dominado por una variable de numero entero que hace que la posición del Servo4 llegue a la posición central en la que está calibrado, como se explica en el apartado 3.6, y también comprueba con la subrutina CPulsos que no se pase el servomotor de la posición central.

4.1.6.3.2.3 Subrutinas para el movimiento de la cámara hacia arriba o abajo

Las subrutinas que mueven la cámara hacia arriba y abajo se llaman **CArriba** o **CAbajo**, sus instrucciones son las mismas que para las subrutinas de **CIzquierda** y **CDerecha**, lo único que cambia es que el servomotor utilizado en este caso es el Servo3, encargado del movimiento vertical de la cámara, y por lo tanto, lo único que varía son las posiciones del Servo3. De esta manera se consigue que la cámara vaya hacia arriba o hacia abajo.

4.1.6.3.2.4 Subrutina para el movimiento de la cámara hacia la posición de reposo

La subrutina que sitúa la cámara en la posición de reposo se llama **CReposo** y se basa en un bucle dominado por una variable de número entero que hace que la posición del Servo4 y del Servo3 llegue a la posición central en la que está calibrado. También comprueba con la subrutina CPulsos que no se pasen los servomotores de la posición central.

4.1.6.3.3 Subrutinas de movimientos del robot

4.1.6.3.3.1 Subrutina para que el robot gire hacia la cámara

La subrutina que hace que el robot gire hacia la posición de la cámara denominada **Camara**, se basa en ir haciendo comparaciones entre distintas posiciones del Servo4. Lo primero que hace es averiguar si la posición del Servo4 es la misma que la posición central ya calibrada, si es así termina la subrutina. Si no se da el caso que la posición del Servo4 es la central, entonces se compara primero si la posición del Servo4 es más grande o más pequeña que la de la posición central ya calibrada. Si se da el caso se debe hacer girar la cámara a la izquierda o a la derecha respectivamente y asegurar que la cámara vuelva a su posición central.

4.1.6.3.3.2 Subrutina para que el robot gire hacia el norte

La subrutina que hace que el robot gire hacia el norte, es en principio la más complicada ya que utiliza la brújula del SR1 previamente calibrada como se explica en el apartado 3.6. Dicha subrutina empieza dándole unos valores de aceleración a los servos, una vez hecho esto la subrutina hace una llamada a la subrutina **Dir**, que es la que se encarga de comprobar la dirección en la que está posicionado el robot, leyendo el valor que le pasa la brújula en grados y quitarle los decimales para obtener un número entero. En ese momento la subrutina gracias al dato en grados de la dirección, comprueba si éste es mayor o menor de 180^{o}, con ello se logra que cuando el robot gire hacia el norte lo haga por el lado más próximo a éste haciendo el menor giro posible.

Una vez se ve a qué dirección ha de girar, comienza un bucle que dura hasta que el robot está orientado hacia el norte. Dentro de dicho bucle se comprueba la dirección constantemente para que no se pase del norte, y una vez está próximo a este, va frenando los servomotores del robot y controlando también que estos no se pasen de su valor máximo y mínimo en ningún momento por precaución. Una vez el robot llega al norte, sale de la subrutina y vuelve al bucle principal.

4.1.6.3.3.3 Subrutina para que el robot vaya hacia delante o hacia atrás

Las subrutinas de marcha adelante o marcha atrás, son llamadas **Adelante** o **Atras**, ponen los dos servomotores de tracción en posición de reposo, entrando en un bucle, que marca la distancia que recorre el robot dependiendo del valor que se le dé a la variable de este. Dentro del bucle vienen las instrucciones para que el servo 1 y 2, que son los servomotores de tracción, avancen o retrocedan y realice una llamada a la subrutina Pulsos, que es la que se encarga de que los servomotores no se pasen de sus valores máximos y mínimos que se calibran como se explica en el apartado 3.6. Si esto ocurre puede dañarlos y hacer que el programa no responda correctamente. La subrutina para que avance más tiempo funciona igual que para el de marcha adelante, lo único que cambia es el valor de la variable del bucle, que hace que esta recorra mas camino.

4.1.6.3.3.4 Subrutina para que el robot realice un beep-beep o un beep

La subrutina llamada **Beep-Beep** asigna un número entero a una variable cuyo valor es 2000, que es la frecuencia de resonancia del altavoz que incorpora el SR1 y por lo tanto es la más ruidosa de todas. A continuación hace un **Call PutPin (Led, Lon)** para encender los leds del SR1, después efectúa el doble Beep mandándole al altavoz la frecuencia de resonancia, por último se apagan los leds del robot y sale de la subrutina.

La subrutina llamada **Beep** se realiza de la misma manera que la anterior, lo único que varía es que no se enciende ningún led y se envía un simple Beep con la misma frecuencia de resonancia.

4.1.6.3.4 Subrutina para que el robot realice la telemetría

La subrutina de telemetría sirve para medir la distancia, la luz, la dirección con la brújula, el rumbo que tiene marcado, la temperatura, si el robot está inclinado o no y si tiene algún paragolpes presionado.

4.1.6.3.4.1 Subrutina para que el robot realice la medición de la distancia

La subrutina que mide la distancia con el sensor SRF08 y lo envía por el puerto serie es llamada **Sonar**, lo que se pretende hacer es la medición en centímetros y esperar 70 ms para que termine, una vez transcurrido ese tiempo lee la distancia medida y limita su valor a 3 caracteres para evitar un error en la recepción. El valor obtenido lo convierte en una cadena de 3 caracteres de longitud fija, para enviarlos por el puerto serie sin problemas en la recepción. Para finalizar manda la cadena y 3 bytes por el puerto serie.

4.1.6.3.4.2 Subrutina para que el robot realice la medición de la luz

La subrutina que mide la luz con el sensor SRF08 y lo envía por el puerto serie es llamada **Luminosidad**, primero se hace la medición en centímetros y espera 70 ms para que se termine, una vez transcurrido ese tiempo lee el sensor de luz y convierte su valor en una cadena de 3 caracteres de longitud fija, para enviarlos por el puerto serie sin problemas en la recepción. Para finalizar manda la cadena y 3 bytes por el puerto serie.

4.1.6.3.4.3 Subrutina para que el robot realice la medición de la dirección

La subrutina que mide la dirección con el sensor brújula y lo asigna a **DirACT** para los giros es llamada **Dir**, lo primero que hace es leer el registro de la dirección en grados. Una vez obtenido dicho valor se le quitan los decimales y el valor obtenido se lo asignaremos a **DirACT**.

4.1.6.3.4.4 Subrutina para que el robot realice la medición de la dirección con el sensor brújula digital

La subrutina que mide la dirección con el sensor brújula digital y lo envía por el puerto serie es llamada **Brujula**, lo primero que hace es leer el registro de la dirección en grados, una vez obtenido dicho valor se le quitan los decimales, y se convierte el valor del rumbo en una cadena de 3 caracteres de longitud fija. Para finalizar manda la cadena y 3 bytes por el puerto serie.

4.1.6.3.4.5 Subrutina para que el robot realice la medición de la temperatura

La subrutina que mide la temperatura con el sensor Ds1820 se llama **Tempera**, lo primero que hace es llamar a las subrutinas **Call Reset1W**, para resetear el bus de un hilo. Realiza un **Call BWrite1W (&HCC)** para mandar el comando de salto de rom y un **Call BWrite1W (&H44)** que es el comando de medir la temperatura. Una vez llamadas a las subrutinas se hace un retardo de 750 ms y se vuelve a llamar a las subrutinas de **Call Reset1w**, **Call BWrite1W (&HCC)** y **Call BWrite1W (&HBE)** que es para el comando del lectura del registro.

Una vez llamadas a las subrutinas lo que se hace es leer el byte de la temperatura y su signo, si el signo es negativo, se le hace complemento a dos para convertirlo. Una vez obtenido el valor correctamente de la temperatura, se convierte en una cadena de 3 caracteres de longitud fija y se manda junto a 3 bytes por el puerto serie.

4.1.6.3.4.6 Subrutina para que el robot compruebe el sensor de inclinación

La subrutina que comprueba el sensor de inclinación se llama **Inclinacion**, lo primero que hace es crear una variable con el valor de inclinación como cadena. Este valor lo pone en formato de 3 bytes y una vez cambiado el formato, comprueba el sensor de inclinación. Pone el valor de la inclinación en formato de 3 bytes de nuevo, para mandar el comando obtenido y 1 byte por el puerto serie.

4.1.6.3.4.7 Subrutina para que el robot compruebe el paragolpes izquierdo o el derecho

Las subrutinas de comprobación del paragolpes izquierdo o derecho son iguales, la única diferencia entre una y otra es que una comprueba el paragolpes izquierdo y la otra el derecho. Lo primero que hace es crear una variable con el valor del paragolpes como cadena. Este valor lo pone en formato de 3 bytes y una vez cambiado el formato, comprueba el paragolpes izquierdo o derecho. Pone el valor del paragolpes en formato de 3 bytes de nuevo, para mandar el comando obtenido y 1 byte por el puerto serie.

En cuanto se termina de explicar las rutinas con las que se programa el robot para que realice los movimientos deseados por el usuario, el usuario debe de enviar los comandos a través del programa SR1Explorer y mediante el puerto serie son captados por el robot previamente programado.

4.1.6.4 Diagrama de bloques del programa

Para finalizar la explicación del programa y proseguir con el programa del PC, la figura 4.4 muestra un diagrama de bloques de cómo funciona el programa y sus distintas subrutinas; ya que facilita enormemente la comprensión del esqueleto del programa y su funcionamiento.

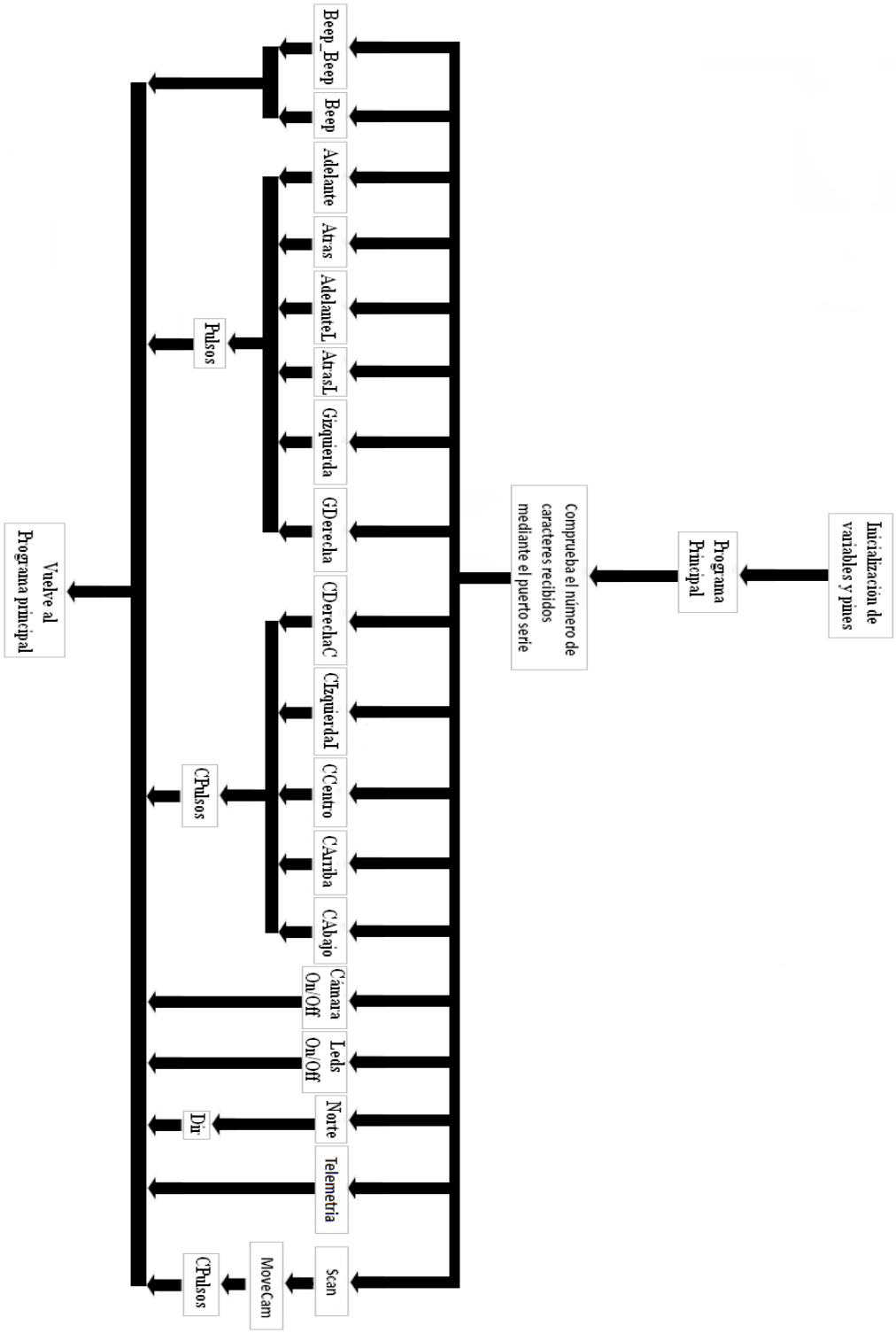


Fig. 4.4 *Cronograma del programa Explorer*

4.1.6.5 Explicación del programa del PC

Con el programa del PC lo que se consigue es poder controlar el robot mediante los botones que se ilustran en la figura 4.3. De esta manera se puede dirigir y controlar las acciones que se desea que el robot reproduzca.

Se ha desglosado el programa en cuatro apartados, en el primero se explica la imagen remota, figura 4.5, en el segundo apartado se dedica tanto en el control de la cámara como en el desplazamiento, figura 4.6, en el tercer apartado se habla del escáner sonar, figura 4.7, y para finalizar se completa la explicación con la telemetría, figura 4.8.

Para activar la cámara y que se pueda ver la imagen, hay que pulsar el botón de **Camara ON**, en el panel de telemetría, una vez pulsado el botón se manda el comando al SR1 para que active la alimentación de la cámara. De forma simultánea, la imagen del robot de la pantalla se abre y se activa la capturadora de video del PC. Si el receptor de video de la cámara está conectado a la capturadora, la imagen aparece en la pantalla. En caso de que no aparezca, es necesario pulsar sobre el botón de Ajustes que se encuentra en el panel de control de la cámara y seleccionar la capturadora si es que hay y se tiene la entrada de video correspondiente.

En caso de que no se disponga de tarjeta capturadora, la imagen se puede ver en un monitor o televisión aparte, para ello es necesario conectar el receptor a la entrada de video de la televisión.

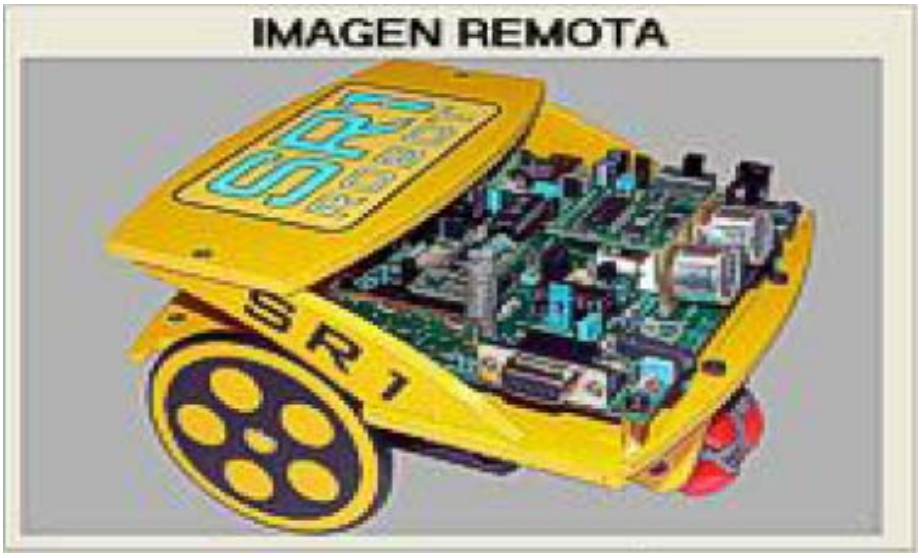


Fig. 4.5 *Imagen remota de la cámara.*

Para desactivar la cámara lo único que hay que hacer es apretar el botón de **Camara OFF**, en el panel de telemetría, y así se corta la alimentación de la cámara y se va cerrando el panel de la imagen de la cámara.

El panel de control de la cámara permite controlar tanto su movimiento, como guardar y recuperar posiciones en diferentes memorias. Al pulsar sobre los botones rojos, se guarda la posición actual de la cámara. Al pulsar en los botones verdes se recupera la posición guardada.

De esta forma se puede mover la cámara entre diferentes puntos de forma rápida y sencilla. El botón de **Ajustes** permite acceder al panel de control de la capturadora de video. El botón **Grabacion** realiza una grabación de la imagen de la cámara en el archivo que se especifica en el panel de control de la capturadora.

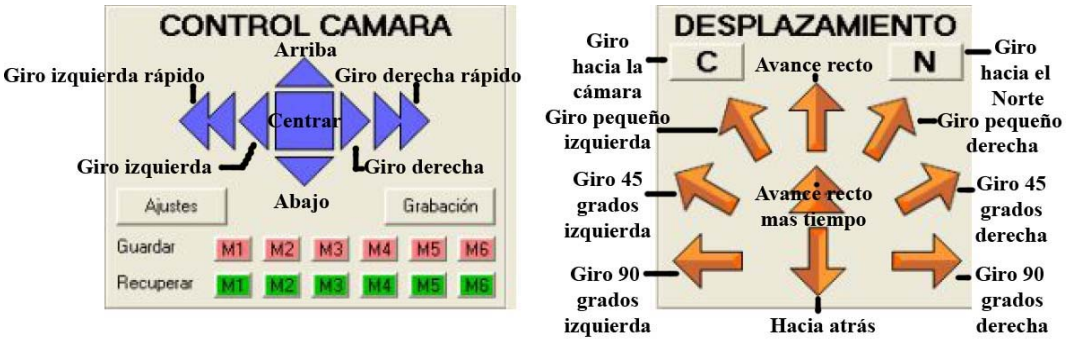


Fig. 4.6 *Controles tanto de la cámara como del robot*

El panel de desplazamiento contiene los controles necesarios para mover el robot en todas direcciones. El control de los giros no está basado en grados de la brújula, ya que este método resulta más lento y por lo tanto poco práctico. El botón **C** hace que el robot gire hacia donde apunta la cámara, a la vez que simultáneamente gira la cámara hasta que se queda en su posición central. En este caso si que se utiliza la brújula digital para calcular y controlar la posición del robot mientras se produce el giro. De forma similar, el botón **N** orienta el robot hacia el norte, basado de nuevo en la información que proporciona la brújula digital.

El escáner, figura 4.7, hace un barrido que permite hacer una representación grafica del entorno. Para poder hacer el barrido, se tiene que colocar el sensor de ultrasonidos en el cabezal móvil de la cámara y así se puede mover tanto en horizontal como en vertical. Se incluyen tres botones que permite seleccionar entre tres alturas de escaneo diferentes, lo que permite tener una visión más precisa de los objetos del entorno. Cada una de las alturas se representa mediante un color diferente, permitiendo de esta forma tener una representación más clara del entorno. El escáner también proporciona una barra gráfica con la información del nivel de luz que rodea al robot. El color de la barra resulta más claro cuanto mayor es la luminosidad.

Fig. 4.7 *Pantalla del barrido del sonar*

Además de la información gráfica, se muestran los valores digitales máximos y mínimos de las mediciones, tanto de la distancia como del nivel de luz. Para finalizar se incluyen botones para cambiar el rango de la gráfica, de forma que se obtenga una visión lo más clara posible de la función de la distancia a la que se encuentren los obstáculos. Las cuatro posibles opciones están prefijadas a los valores de 50, 75, 150 y 450 centímetros de fondo de escala.

En el panel de telemetría se puede apreciar que es donde apretando uno a uno los botones de medidas, se ven las mediciones realizadas de la distancia a la que tenemos un obstáculo, la luminosidad que hay en ese momento, la temperatura que hace en el ambiente, el rumbo marcado por la brújula. Se marca en la casilla de inclinación, bumper izquierdo y bumper derecho activado si el robot está inclinado o si alguno de los dos bumpers está presionado. A parte de hacer las medidas en este panel se puede activar y desactivar tanto la cámara como los leds. Se puede realizar un tono simple por el altavoz o si se quiere dar un aviso, apretando el botón alarma se oyen dos bips y se encienden y apagan los leds. El botón de telemetría permite realizar todas las mediciones a la vez con pequeños retardos entre sí, para así quitar el engorro de ir una a una.

Fig. 4.8 *Panel de telemetría*

4.1.6.6 Detección de errores en el programa

A la hora de ejecutar el programa y empezar a controlar el robot mediante el ordenador. Se observa que el robot no realiza correctamente las órdenes que se le dan. Los problemas que se han detectado son los siguientes:

- Cuando se le da la orden de girar el robot a la derecha, en vez de girar a la derecha gira hacia la izquierda y viceversa.

- Cuando se le da la orden de girar la cámara a la derecha, sucede lo mismo, en vez de girar a la derecha gira hacia la izquierda y viceversa.

- Cuando se quiere subir la cámara sube, pero al intentar hacer que baje, la cámara no baja, todo lo contrario, continúa subiendo.

-Cuando se soluciona este problema, surge otro. La cámara cuando se quiere que suba, baja y cuando se quiere que baje, sube.

- Cuando se intenta girar el robot hacia la dirección de la cámara, el robot gira hacia donde la cámara mira, pero la cámara no vuelve a su posición central.

- Cuando se quiere girar hacia el norte, después de calibrar bien la cámara como se explica en el apartado 3.6, el robot solo gira por el camino más largo, nunca hace el giro más óptimo.

- Cuando se quiere hacer el barrido con el sonar de ultrasonidos, siempre hace el barrido bajo por culpa de que el servomotor vertical de la cámara no funciona correctamente.

A cada uno de estos problemas se encuentran sus soluciones que están explicadas en el apartado 5.3 como modificación del programa EXPLORER.

4.1.6.7 Valoración del Explorer

En este programa, tal y como se explica en el apartado anterior, tiene ciertos errores que provocan que el robot responda mal y no ejecute correctamente las órdenes que se le envían desde el ordenador. Una vez solucionado todos los fallos, el robot responde al 100% obteniendo unos resultados buenos y precisos.

4.1.7 Programa Navegador de Líneas

4.1.7.1 Explicación del programa NAVLINEA

El programa NAVLINEA hace que el robot localice y siga el recorrido marcado por una línea sin perderla y haga el recorrido en el menor tiempo posible.

Para el seguimiento de la línea se emplea un sensor especial que se coloca en la parte trasera del SR1 suspendido 1 cm del suelo. El sensor, figura 4.9, está formado por tres emisores y tres receptores de infrarrojos, separados entre sí unos 25 mm. Cada sensor emite unas señales de infrarrojos que se reflejan en el suelo y que son recibidas por el receptor que es en realidad un fotodiodo que entra en conducción cuando recibe la luz. Cuando no recibe la luz infrarroja procedente del suelo, como por ejemplo cuando detecta la línea negra, el fotodiodo no conduce y la salida correspondiente cambia de estado.

El sensor también incluye tres diodos leds que actúan mostrando el estado de los sensores de infrarrojos. Mientras se detecta el suelo, el diodo permanece encendido, y se apaga cuando detecta la línea negra. Esto facilita el trabajo, ya que para saber si realmente hace bien su tarea de detectar la pista, solo se tiene que ver si los leds se encienden o apagan, y por lo tanto, si detectan bien la pista o no. En caso de no detectarla, encontrar su fallo y arreglarlo, y sobretodo, llegar a mejorar los tiempos hechos por el robot en el circuito.

Fig. 4.9 *Sensor formado por tres emisores y tres receptores de infrarrojos*

El sensor se coloca en la parte trasera del SR1, como se muestra en la figura 4.10, lo que hace que el robot vaya en sentido contrario. Se ha puesto en la parte trasera del robot, junto a las ruedas de giro, porque así se puede llegar a hacer el recorrido con más precisión, ya que si se sale de la pista pueda rectificar lo antes posible. Si se pone en la parte delantera con la rueda omnidireccional, entonces si se sale de la pista no le da tiempo a rectificar, es demasiado tarde y el giro se hace fuera del recorrido.

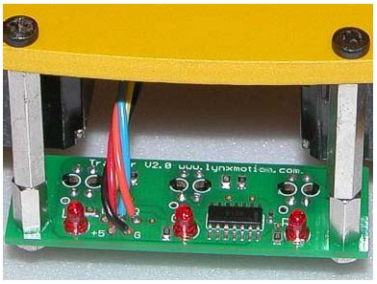


Fig. 4.10 *Seguidor de líneas colocado en la parte trasera del SR1*

El procedimiento consiste en hacer una lectura de los sensores, y en función de los datos obtenidos hacer una u otra acción. Se convierten las señales procedentes de los 3 sensores en un número que representa el valor binario de los tres sensores. Dado que son tres sensores y que cada uno puede tener solo dos estados, encendido o apagado, el total de combinaciones posibles es 2 elevado a 3. Por lo que si se convierte el valor binario de los 3 sensores en un número natural, se obtiene un número comprendido entre el 0 y el 7.

Comparando el valor de este número con uno de los ochos posibles valores se tienen cubiertas todas las posibilidades de los sensores de infrarrojos. Una vez leído el estado de uno de los sensores, lo que hace es guardarlo en la variable Status y luego multiplicarlo por 2. El efecto de la multiplicación es el mismo que desplazar cada uno de los 8 bits que tiene el byte una posición a la derecha, con lo que la posición del bit que se encuentra más a la derecha se queda lista para recibir el nuevo valor del siguiente sensor mediante la operación suma.

Una vez efectuada la llamada a la función, se tiene una variable llamada Status que representa el estado actual de los tres sensores de infrarrojos. Lo único que hay que hacer es determinar la acción a realizar en función del valor de la variable. Como se ve en la tabla, hay situaciones como en el caso del valor 5 que corresponde a 101 en los sensores, en este caso los dos sensores extremos están detectando el suelo y el central está apagado por que está detectando la línea negra. En esta situación el robot está siguiendo perfectamente la línea sin perder su rastro.

El caso 2 en teoría no debería de ocurrir, ya que implicaría un fallo de detección de la pista, que se salga fuera de la línea y se encuentre con un borde, una curva muy cerrada, o simplemente un reflejo erróneo del sensor causado por cualquier causa, pero se tiene que tener en cuenta. En este caso ya que se trata de situaciones puntuales, lo mejor es que siga recto y vaya repitiendo el ciclo hasta que se encuentren valores normales que si se correspondan con algunas de las situaciones probables.

Los casos 1, 3, 4 y 6 son los casos probables en los que se produce un desvío de línea e incluso se puede distinguir entre desvío ligero y desvío grande. El caso 5 es la situación normal y la que se pretende realizar en este programa, que no pierda la línea y siga su pista perfectamente.

El caso 7 es una excepción que se produce cuando el robot ha perdido todo contacto con la línea. En este caso lo mejor es avanzar lentamente, hasta que se vuelva a detectar otra vez la línea ya que si se hace muy rápido no se hace el giro a la suficiente velocidad y se vuelve a perder la línea.

Decimal	Binario	Significado
0	000	Línea detectada transversalmente. Lo mejor es seguir con la acción que se estaba haciendo hasta que se corrija
1	001	Se desvía hacia la derecha. Hay que avanzar hacia la izquierda
2	010	No debería ocurrir. Se sigue recto hasta que cambie
3	011	Se desvía mucho hacia la derecha. Hay que girar hacia la izquierda
4	100	Se desvía hacia la izquierda. Hay que avanzar hacia la derecha
5	101	Camino correcto. Sigue recto
6	110	Se desvía mucho hacia la izquierda. Hay que girar hacia la derecha
7	111	No se detecta la línea. Avanzar lentamente hasta que se detecte el borde de la línea

Tabla. *Tabla con las distintas situaciones que pueden suceder*

4.1.7.2 Diagrama de bloques del programa

Para finalizar la explicación del programa, se muestra en la figura 4.4, un diagrama de bloques de cómo funciona el programa y sus distintas subrutinas; ya que facilita enormemente la comprensión del esqueleto del programa y su funcionamiento.

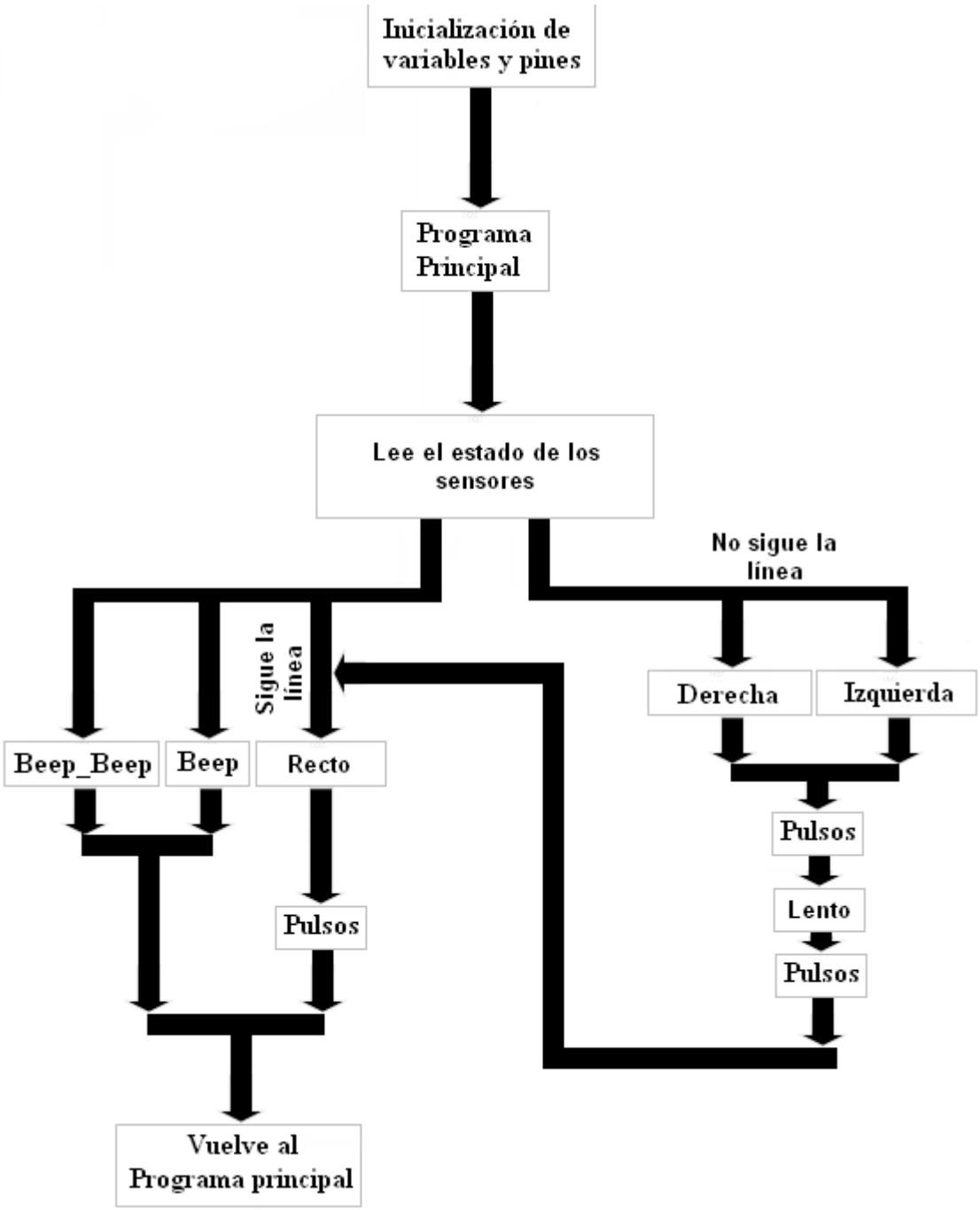


Fig.4.11 *Cronograma del programa NAVLINEA*

4.1.6.7 Valoración del NAVLINEA

Para ver el correcto funcionamiento de este programa se realiza un circuito en un tablero, de ciertas dimensiones como se muestra en la figura 4.12. De esta manera se puede observar si hay que modificar el programa, para que el robot realice el circuito con el menor tiempo posible o si ya se piensa que está bien optimizado. Desde nuestro punto de vista, el robot está bien optimizado, realizando el circuito en un tiempo bueno y sin salirse de él.

Se comprueban los casos citados en la tabla, ya que hay momentos en los que el robot pierde la línea e intenta recuperarla, hay otros momentos en los que el robot encuentra una curva muy cerrada. En esta situación se crea un bucle en el robot y no puede salir de él.

Se han grabado unos videos del robot en un circuito realizado por nosotros, que se pueden encontrar en el CD, denominados NAVLINEA1 y NAVLINEA2. En estos videos se observa como el robot recorre el circuito de la figura 4.12 y como el seguidor de líneas detecta la pista, encendiendo y apagando los leds.

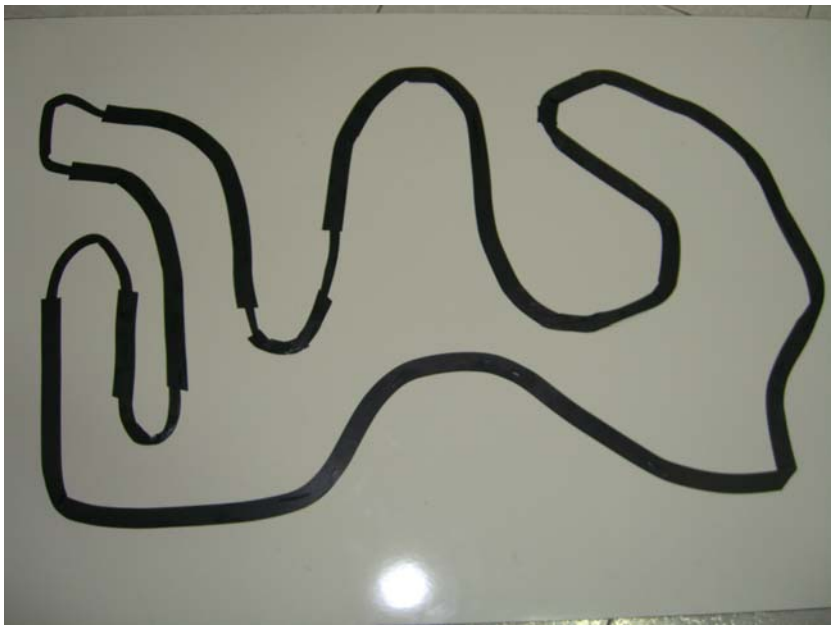


Fig. 4.12 *Circuito Seguidor de Líneas*

5. Aplicaciones realizadas

En este tema se pueden ver los programas **"Control Remoto SonyIR"** y **"Dirección Norte"**, los cuales han sido hechos partiendo desde cero, y la modificación del programa **"Explorer"**.

Los dos programas tienen finalidades distintas, ya que mientras el primero tiene como objetivo el control del robot por parte del usuario, el segundo hace que el robot sea autosuficiente y siga una trayectoria hacia el Norte sin desviarse de él, esquivando obstáculos y tomando una imagen de estos.

En cuanto a la remodelación del programa **"Explorer"**, la programación errónea en varias partes de su código hacía que no funcionaran más del 50% de sus funciones, por lo que tuvo que ser remodelado casi por completo para su correcto funcionamiento.

5.1. CONTROL REMOTO SONYIR

5.1.1. Explicación general

El programa CONTROL REMOTO SONYIR, hace que el robot obedezca las órdenes que le se le dan mediante una señal de infrarrojo recibida desde un mando a distancia Sony.

El primer paso a seguir, es hacer un puente entre el conector J6 y el J8 para que la señal del sensor de infrarrojo quede conectada al pin 12 del BasicX24, de esta forma se consigue que el robot reciba por el receptor de infrarrojos las instrucciones que el usuario manda a través del mando a distancia.

Una vez seguidos los pasos básicos de preparación de la parte de hardware, se pasan a enumerar las funciones de las que se ha dotado al programa.

5.1.2. Explicación del programa principal

En primer lugar se dispone del programa principal, el cual se basa prácticamente en recibir los datos que capta el receptor de infrarrojos, mandados por el control remoto y transformarlos en un número entero, el cual es interpretado por una orden por el robot. Dicho proceso se explica en el apartado 5.1.6, en el que se aclaran varios aspectos del programa.

La función del programa es esperar a detectar un impulso de al menos 2 ms, lo cual significa que se ha detectado el impulso de inicio de un código de infrarrojos. A continuación inicia el proceso de captura de los 24 impulsos correspondientes a los 12 bits que corresponden a la orden. Hay que destacar que a la hora de evaluar los impulsos, se hace de dos en dos, es decir los impulsos pares, ya que los impares corresponden a los impulsos altos de separación entre cada bit, que no contienen información.

Es importante esperar a recibir el impulso de inicio antes de llamar a **'InputCapture'**, ya que si se llama a este procedimiento y no se produce ningún impulso el programa se queda esperando indefinidamente, tal y como se ve en la explicación 5.1.7.

También hay que comprobar que el impulso inicial tiene cierta duración, ya que el sensor de infrarrojos es muy sensible y no es extraño que de vez en cuando mande pequeños impulsos de tan solo unos cuantos microsegundos como consecuencia de interferencias o cambios lumínicos, lo que podría disparar el proceso de captura, obteniendo datos erróneos.

Antes de transformar el dato, el programa principal se encarga de inicializar el robot, asegurándose de que tanto los servos de la cámara como los del robot estén en su posición central y parados, que la cámara este desactivada, los leds apagados y espera a recibir un impulso del receptor de infrarrojos.

Un vez recibido el impulso el programa comienza a funcionar, primero comprueba si es el inicio de un código IR lo que ha recibido, y si es así empieza a comprobar todos los pulsos recibidos transformándolos en segundos. Una vez hecha esta operación guarda el valor obtenido con forma de numero entero y llama a una subrutina llamada "**MandoIR",** para que se encargue de ver qué tipo de tecla se ha pulsado en el mando a distancia y que operación ha de realizar. Dicha subrutina comprueba cual de las siguientes instrucciones ha de realizar el robot: Robot adelante y atrás mucho o poco recorrido, giro a la izquierda y derecha 15, 45, 90 y 135 grados; girar la cámara arriba, abajo, izquierda, derecha y posición central; encender y apagar cámara y leds; girar hacia el norte; y hacer uno o dos Beeps dependiendo del botón pulsado.

Cada una de las opciones tiene, como es normal, asignada su subrutina, las cuales son explicadas detenidamente.

5.1.3. Explicación de las subrutinas

5.1.3.1 Las subrutinas “Adelante” y “AdelanteL”

En el caso de que el comando IR sea el de marcha adelante, el programa llama a la subrutina llamada **“Adelante”**. Dicha subrutina pone en principio los dos servos en posición de reposo, entrando en un bucle, el cual marca la distancia que recorre el robot dependiendo del valor que se le dé a la variable, siendo esta la que determina las veces que se repite el bucle. Dentro del bucle están las instrucciones para el servo 1 y 2, los cuales se ocupan de que el robot se mueva hacia delante, y una llamada a la subrutina **“pulsos”**, que es la encargada de que los servos no pasen de sus valores máximos y mínimos que se han calibrado en el proceso de calibración del robot(Ver apartado 3.5.1), ya que si esto ocurriera podría dañarlos y hacer que el programa no respondiera correctamente. La subrutina **“AdelanteL”** actúa de igual forma que **“Adelante”**, con la única diferencia de que el robot avanza menos distancia con **“AdelanteL”**, esta subrutina sirve para que el robot pueda maniobrar correctamente en lugares que requieren una gran precisión. En el código del programa de la subrutina, la única diferencia radica en el valor de la variable del bucle, haciendo que esta recorra menos camino al asignarle un número entero menor que a **“Adelante”.**

5.1.3.2 Las subrutinas “Atras” y “AtrasL”

Las subrutina **“Atras”** es la que se encarga de que el robot se mueva hacia atrás. Esta subrutina funciona prácticamente igual que la de marcha adelante, con el único matiz de que dentro del bucle que determina la distancia recorrida por el robot, las órdenes dadas a los servos son dé marcha atrás en vez de marcha adelante. Lo mismo ocurre con la subrutina **“AtrasL”**, que es la que se encarga de que el robot se mueva marcha atrás una distancia menor que con la subrutina **“Atras”**, siendo la única diferencia entre subrutinas el valor de la variable del bucle, la cual ha de ser menor para que recorra una distancia más pequeña.

5.1.3.3 Las subrutinas “GIzquierda” y “GDerecha”

Las subrutinas **“GIzquierda”** y **“GDerecha”** son las que se encargan de que el robot haga giros de 5, 45, 90 y 135 grados hacia la izquierda y derecha respectivamente. La precisión del giro de ambas subrutinas es un tema complejo, ya que dependen de muchos factores, como son el estado de las baterías o el pavimento por el que se mueva el robot, pudiendo estas condiciones variar ligeramente los grados del giro. Hecha ya esta aclaración se pasa a explicar cómo se comporta el programa cuando se le aplica esta instrucción y como se ha realizado.

En primer lugar el programa comprueba el botón que ha sido pulsado del mando a distancia, ya que cada uno tiene asignados unos grados a girar, los cuales dependiendo del código IR que lea, manda un número entero a la variable **GIzquierda** o **GDerecha**, dependiendo de la dirección del giro que se quiere efectuar. Como ambas subrutinas son prácticamente iguales y solo cambia la dirección a la que se mueven los servos, ambas subrutinas son explicadas a la vez.

Una vez le ha sido enviado el grado del giro a la subrutina, esta lo guarda en una variable llamada **Grados** y se establecen dos valores de aceleración para los servos, en ese momento comienza un bucle, el cual se encarga de hacer girar el servo hacia la izquierda o derecha (depende de la variable) y en el que el número de veces que se repite este viene dado por la variable **Grados**, girando más o menos dependiendo de esta.

5.1.3.4 Las subrutinas “CArriba” y “CAbajo”

Las subrutinas de subir o bajar la cámara funciona de la siguiente manera, al recibir el comando IR este llama a la subrutina **“CArriba”** o **“CAbajo”** dependiendo del botón pulsado. Dicha subrutina no es otra cosa que un bucle, el cual es dominado por una variable entera fija que se encarga de la velocidad a la que gira la cámara. Dentro de este bucle radica la diferencia entre las subrutinas **“CArriba”** y **“CAbajo”**, ya que la única operación realizada es la de sumar o restar valores a la posición del Servo3 (servo que se encarga exclusivamente del movimiento vertical de la cámara) dependiendo de si queremos subir o bajar esta. Una vez hemos movido el servo, se hace una llamada a la subrutina **“CPulsos”** para ver si se ha superado la posición máxima o mínima del servo, y en caso afirmativo, volverlo a colocar en un rango posible.

5.1.3.5 Las subrutinas “CIzquierdaI” y “CDerechaD”

Las subrutinas **“CIzquierdaI”** y **“CDerechaD”** son las encargadas de girar la cámara a izquierda o derecha respectivamente. Ambas subrutinas son prácticamente iguales a lo explicado con las anteriores subrutinas **“CArriba”** y **“CAbajo”**, la única diferencia radica en que el servo utilizado en este caso es el servo 4 que es el que se encarga del movimiento horizontal, y por lo tanto, lo único que varia es que en el bucle de la subrutina se varían las posiciones del Servo4 dependiendo de si se quiere que la cámara gire a la izquierda o a la derecha.

5.1.3.6 La subrutina “CCentro”

La subrutina de girar la cámara a su posición central es llamada **“CCentro”**, la cual se basa en un bucle dominado por una variable de numero entero, que hace que la posición del servo 4 llegue a la posición central en la que esta calibrado el servo 4. La subrutina también comprueba mediante la subrutina **“CPulsos”** que no se pase el servo de la posición central.

5.1.3.7 El método cámara y leds On/Off

En el caso de encender y apagar cámara y encender y apagar leds no se hace uso de ninguna subrutina, y se ha implementado de la siguiente manera para solamente usar un único botón del mando a distancia para encender y apagar.

Para el caso de encender y apagar cámara se ha implementado de la siguiente manera:

Primero el programa comprueba si el usuario ha pulsado el botón de encender/apagar cámara, en caso afirmativo el programa aplica un retardo realizado mediante un bucle for, ya que al pulsar el botón es muy posible que si no se hace muy rápido, se le estén mandando varios pulsos al robot y este esté encendiendo y apagando la cámara constantemente, de esta manera se asegura de que solo llegue un pulso ya sea para encender o apagar.

Acto seguido comprueba si se ha pulsado con anterioridad el botón para saber si el pulso enviado es para encender o apagar cámara, este paso es realizado mediante una variable, la cual si está con el valor entero 0 significara que no ha sido pulsado y si está a 1 es que la cámara ya está encendida. Si la variable está a 0 el método hace un **PutPin(Evc, 0)**, que no es otra cosa que encender la cámara y pone la variable a 1 para indicar que la cámara está encendida, si la variable está a 1 el método hace un **Call PutPin(Evc, 1)** para apagar la cámara y se coloca la variable a 0 para indicar que la cámara está apagada.

El método para encender y apagar los leds es igual que el de encender y apagar la cámara, con la única diferencia que en vez de hacer un **Call PutPin(Evc, 0)** el método hace un **Call PutPin(Led, 0)** para encender los leds y en vez de hacer un **Call PutPin(Evc, 1)** el método hace un **Call PutPin(Led, 1)** para apagar los leds.

5.1.3.8 La subrutina “Norte”

La subrutina girar hacia el norte es llamada **“Norte”** y es la más complicada a priori, ya que utiliza la brújula del SR1 previamente calibrada en el proceso de montaje (Ver apartado 3.5.7). Dicha subrutina comienza dándole unos valores de aceleración a los servos. Una vez hecho esto, la subrutina hace una llamada a la subrutina **“Dir”**, que es la que se encarga de comprobar la dirección en la que esta posicionado el robot leyendo el valor que le pasa la brújula en grados, y quitarle los decimales para obtener un número entero. En ese momento, la subrutina gracias al dato en grados de la dirección comprueba si este es mayor o menor de 180º, con ello se logra que cuando el robot gire hacia el norte lo haga por el lado más próximo a este haciendo el menor giro posible.

Una vez se ha visto a qué dirección ha de girar el robot, comienza un bucle que dura hasta que el robot este orientado hacia el norte, dentro de dicho bucle se comprueba la dirección constantemente para que el giro no se pase de donde está ubicado el norte y, una vez el robot está próximo a este, va frenando los servomotores del robot y controlando también que estos no se pasen de su valor máximo y mínimo en ningún momento por precaución. Una vez el robot ha llegado al norte sale de la subrutina y vuelve al bucle principal.

5.1.3.9 Las subrutinas “Beep” y “Beep_Beep”

Las últimas subrutinas de las que consta el programa son las llamadas **“Beep_Beep”** y **“Beep”**. La única diferencia entre las dos es que la primera efectúa dos tonos y la primera un único tono, por lo que se explican ambas subrutinas de forma general. En primer lugar las subrutinas asignan un número entero a una variable cuyo valor es 2000, que es la frecuencia de resonancia del altavoz que incorpora el SR1 y por lo tanto es la más ruidosa de todas, a continuación hace un **Call PutPin(Led, Lon)** para encender los leds del SR1, después efectúa el primer Beep mandándole al altavoz la frecuencia de 2000 y, si es la subrutina de Beep_Beep repite este paso otra vez para efectuar el segundo tono, por último la subrutina apaga los leds del robot y sale de la subrutina.

5.1.4 Disposición de las teclas del mando IR

Una vez se han explicado las características del programa y sus diversas subrutinas, se muestran a continuación, mediante la figura 5.1 la disposición de las teclas del control remoto Sony IR:

Fig. 5.1 *Usos de las teclas del mando a distancia.*

5.1.5 Números enteros asociados a las teclas del mando IR

Como se ha explicado al principio del programa, cada tecla del control remoto IR es asociada mediante un número entero que lee el programa que a la vez lo asocia a una subrutina, la disposición de los números enteros respecto a las teclas es la mostrada en la siguiente tabla.

Nombre de la tecla	Número entero asociado
Tecla 1	128
Tecla 2	129
Tecla 3	130
Tecla 4	131
Tecla 5	132
Tecla 6	133
Tecla 7	134
Tecla 8	135
Tecla 9	136
Tecla Mute	148
Tecla On/Off	149
Tecla posición pantalla	61
Tecla OK	229
Tecla Menú	224
Tecla Arriba	205
Tecla Abajo	207
Tecla Izquierda	204
Tecla derecha	206
Tecla subir volumen	146
Tecla bajar volumen	147
Tecla subir canal	144
Tecla bajar canal	145

Tabla

5.1.6 Proceso de decodificación del número entero

Antes se ha visto cómo funciona el programa en su totalidad y se ha realizado una explicación de el uso del control remoto, para finalizar la explicación de este programa se termina por explicar como el SR1 transforma la señal que recibe del control remoto en los número enteros expuestos en la hoja anterior.

En primer lugar el control remoto manda un pulso inicial de 2,4 ms que se usa para indicar el comienzo de la transmisión, y a continuación manda un pulso de 12 bits que contienen la información que se quiere pasar.

Cada bit está formado por un pulso alto de 0,6 ms seguido de un pulso bajo que es también de 0,6 ms cuando se trata de un 0 y de 1,2 ms cuando se trata de un 1. Un aspecto importante es que el orden de los bits transmitidos van de menor a mayor, es decir, que el bit de menor peso es el que se envía primero y el de mayor peso el que se envía el último. A continuación se puede ver esta explicación con más claridad si se observa la figura 5.2, en la que se pueden ver las diferentes partes de la señal.

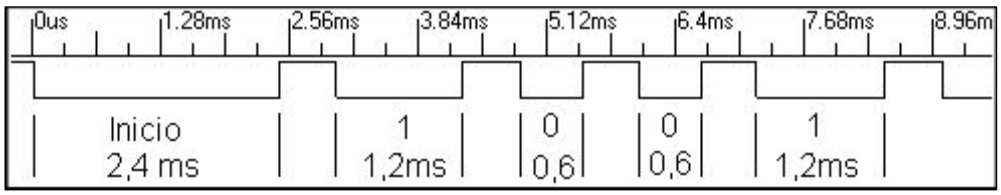


Fig.5.2 *Esquema de una señal IR.*

Se puede apreciar que cuando se pulsa una tecla del mando a distancia, este manda una serie de impulsos luminosos, consistentes en un primer impulso de 2,4 ms, seguido de 25 impulsos cuya duración varía entre los 0,6 si son ceros y los 1,2 ms si son unos. Con estos datos, el tiempo máximo que se tarda en transmitir una orden es de 2,4 + (12 x 1,8)= 24 ms en el caso de que se transmitiera una orden formada por 12 bits a 1, aunque en la práctica la mayoría de las ordenes no ocupan más de 20 ms, por lo que la velocidad resulta suficientemente rápida para transmitir órdenes de forma manual.
Si se mantiene pulsada la tecla del mando a distancia, entonces se transmite una y otra vez la misma orden de forma repetida. La cadencia entre órdenes es de 25 ms, lo que equivale al tiempo máximo que se tarda en enviar una posible orden, como se puede ver en la figura 5.3 en el que se mandan dos señales consecutivas.

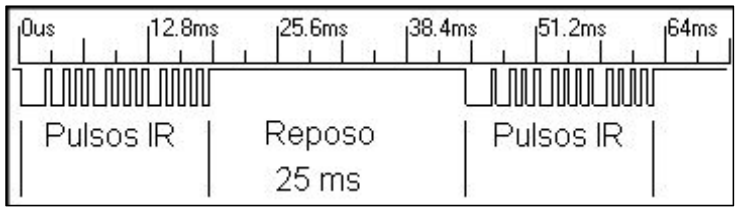


Fig.5.3 *Esquema de dos señales mandadas de forma consecutiva*

5.1.7 El comando '*InputCapture*'

La clave para recibir las señales de infrarrojos en el robot está en la utilización del comando **'*InputCapture*'**, el cual sirve para capturar un tren de impulsos y guardar en una matriz la anchura de los diferentes impulsos capturados.
Al comando también se le debe de indicar el número de impulsos que va a recibir y el tipo de flanco que inicia el proceso de captura, que puede ser flanco ascendente o descendente.
Lo más importante que hay que tener en cuenta al utilizar esta función es:

1. La función no tiene tiempo límite de espera, por lo que la función no vuelve si no se produce por lo memos el impulso inicial. Esto puede hacer que el procesador se quede esperando un impulso de forma indefinida.
2. Una vez se ha producido el primer impulso, si el ancho del impulso es mayor que el rango permitido (8,86ms), se devuelve el valor 65535.
3. Esta función utiliza los temporizadores interno del BasicX24, por lo que el timer 1 no puede emplearse cuando se utiliza esta función. Igualmente se utilizan componentes internos del procesador para hacer la medida de los impulsos. Este hardware está conectado internamente al pin 12 del BasicX24, por lo que es necesario utilizar este pin como receptor y ponerlo en modo de entrada antes de llamar al procedimiento.

5.1.8 Cronograma de las teclas del mando IR

En este apartado se ve cómo queda el cronograma de varias de las teclas del mando a distancia, como se puede apreciar en la figura 5.4.

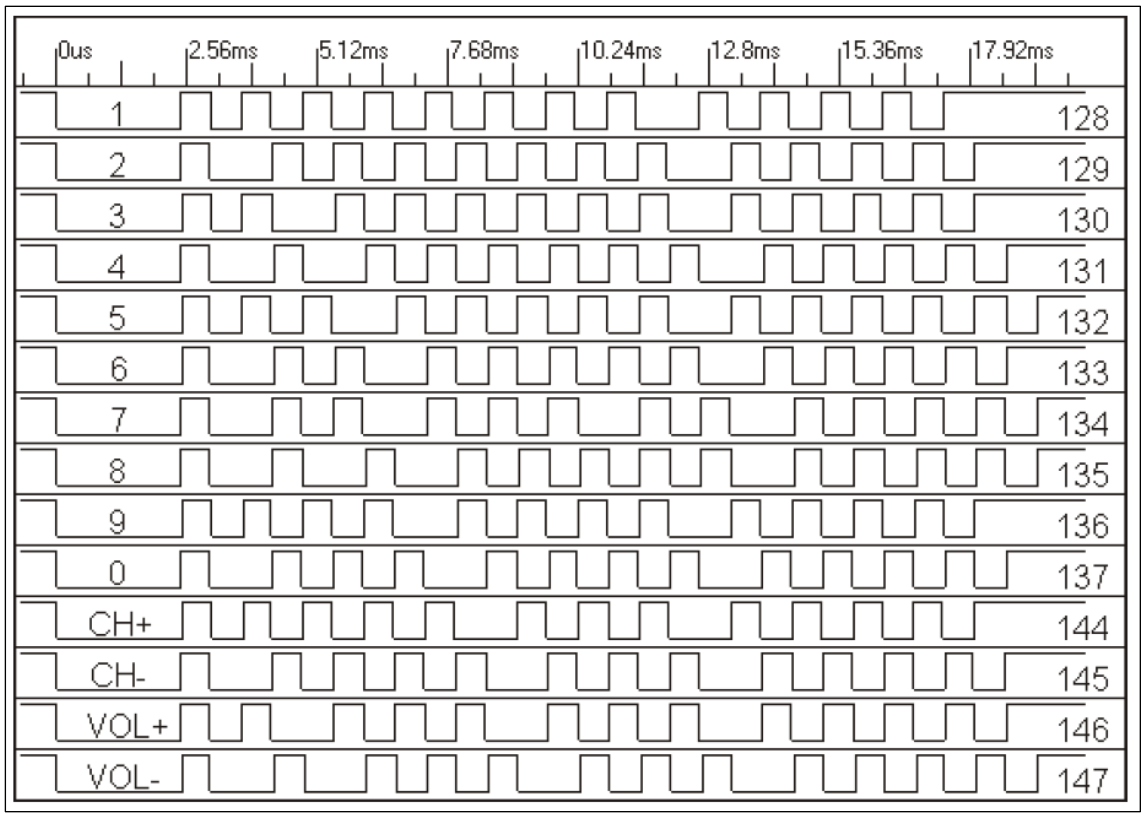


Fig.5.4 *Cronograma de las teclas del mando a distancia*

5.1.9 Diagrama de bloques del programa

Para finalizar la explicación del programa, se muestra un diagrama de bloques de cómo funciona el programa y sus distintas subrutinas, ya que facilita enormemente la comprensión del esqueleto del programa y su funcionamiento como se puede ver en la figura 5.5.

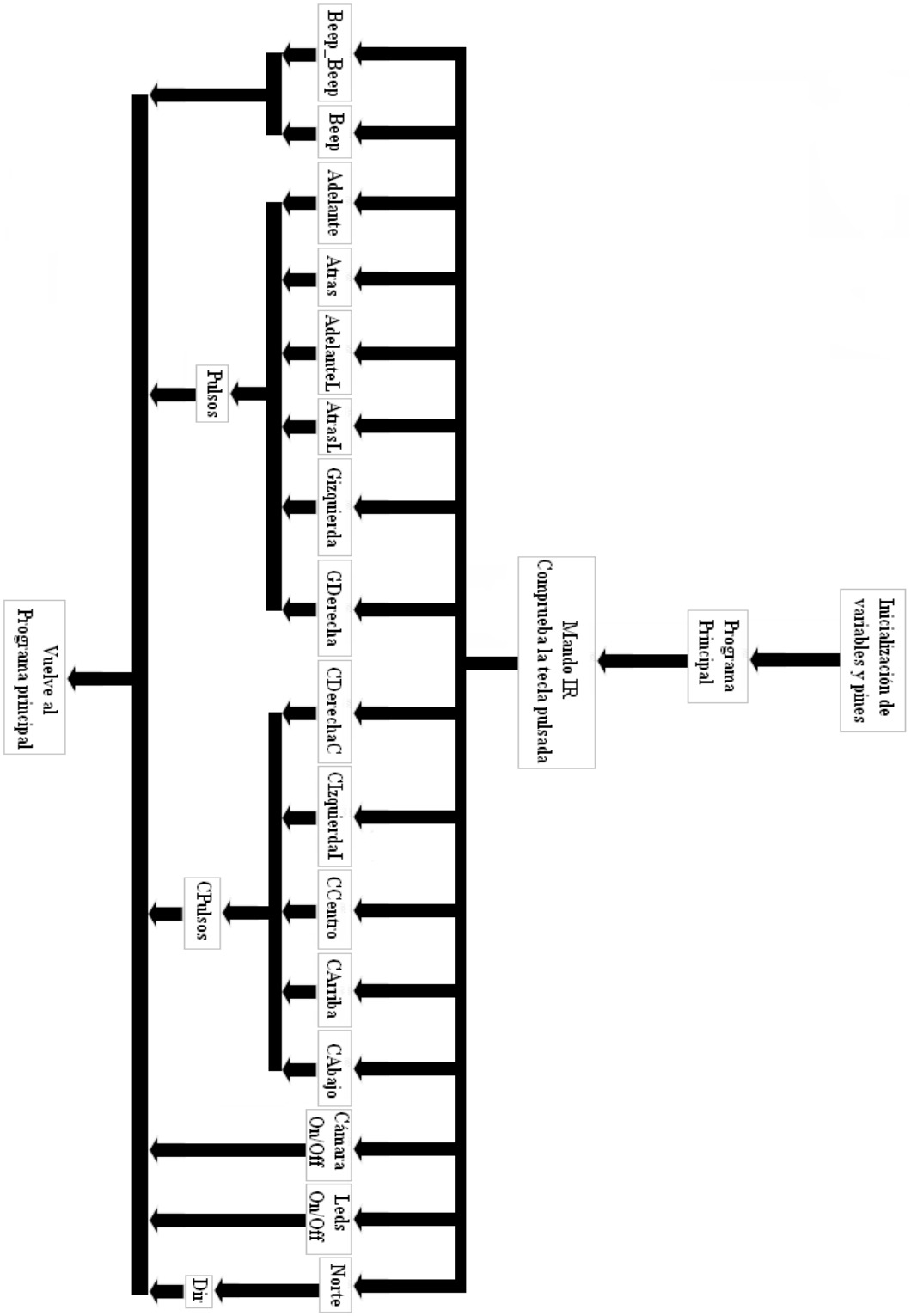


Fig.5.5 *Diagrama de bloques del programa "Control Remoto SonyIR"*

5.1.10 valoración del programa “Control Remoto SonyIR”

Este programa ha sido mejorado constantemente desde que fue creado, la primera versión del programa tenia las funciones de avanzar, retroceder, y girar a los lados. Una vez que todas las aplicaciones funcionaban correctamente se pasó a implementar la parte de girar a izquierda y derecha unos ángulos determinados, en esta parte se tuvieron muchos fallos dependiendo de las baterías del robot y del pavimento donde se realizaban las pruebas, por lo que fue difícil llegar a calibrar todos los ángulos de una manera óptima. A continuación se aplicaron los movimientos de girar la cámara a todas las direcciones y se realizaron los métodos de encender y apagar los leds y la cámara, teniendo estos al principio algunos fallos hasta que aplicamos el retardo entre señales de infrarrojos. Para finalizar se implementó la parte de girar hacia el norte, los movimientos de corto recorrido de adelante y atrás y la alarma del robot.

En su versión final, el programa funciona de manera eficiente, respondiendo este perfectamente a las órdenes que el usuario le da mediante el control remoto. En distancias de aproximadamente 5 metros, las órdenes del mando pueden no llegar al robot, ya que el receptor de infrarrojos no está preparado para captar señales a distancias largas.

En cuanto al resto de funciones, la única que presenta un pequeño defecto es la visualización de imágenes mediante la cámara cuando el robot está en movimiento, ya que la alimentación de los servos y la de la cámara son la misma, por lo que al moverse el robot la cámara no recibe la potencia suficiente para emitir de forma precisa y se aprecian interferencias en la imagen que se visualiza en la televisión.

En definitiva, el programa cumple con creces su objetivo primordial de conseguir mover el robot de forma eficiente con un control remoto Sony, siendo el control del robot sencillo y fiable.

En el CD adjunto a la memoria se encuentra en la carpeta de programas, el código del programa bien explicado y detallado en formato pdf para aclarar el programa realizado.

5.2 DIRECCIÓN NORTE

5.2.1 Explicación general

El programa Dirección norte, hace que el robot SR1 siga una trayectoria de dirección hacia el norte hasta que se encuentra un obstáculo en su camino, en ese mismo momento gira hasta que el sensor de ultrasonidos deja de detectar el obstáculo. Una vez ha ocurrido esto, el robot hace girar la cámara hacia el lugar donde está situado el obstáculo y la enciende un breve periodo de tiempo. De esta manera, se puede visualizar en un televisor o en un ordenador el objeto que ha esquivado el robot.

Pasado el periodo de tiempo, el robot apaga la cámara, la vuelve a colocar en su posición central, y avanza una distancia determinada para poder evitar el obstáculo. Cuando ha terminado esta tarea, el robot vuelve a buscar el norte girando en la dirección más próxima a donde está situado este y sigue avanzando.

5.2.2 Explicación del programa principal

Después de haber explicado el objetivo inicial del programa, se explica cómo funciona el programa principal y sus subrutinas, para que se puedan entender perfectamente los pasos que se han seguido para realizarlo.

En primer lugar se inicializan todas las variables necesarias tanto en el programa principal y en las subrutinas, a su vez se le asignan los pines del robot que tienen que estar conectados o desconectados, para de esta manera poder usar las características que se necesiten de este. Por último se crean las variables para guardar el valor de los servos.

Una vez se ha terminado con la puesta a punto, empieza el programa principal, el cual comienza recuperando los valores de parada de los servos y asignándole a una variable el valor máximo al que se quiere que el sensor de ultrasonidos detecte un obstáculo.

Al realizar diversas pruebas, se llegó a la conclusión que una distancia de 30 centímetros es la óptima para que el error de medida sea mínimo y que el robot le dé tiempo de esquivar el obstáculo. Después de realizar esta tarea, el programa inicializa los servos para que estén en su posición de reposo, se asegura de que los leds del robot estén apagados y se hace una llamada a la subrutina **“Beep_Beep”**, emitiendo esta un pitido de inicio para saber que el programa va a comenzar a hacer su función.

Antes de que el programa comience con su bucle principal, se hace una llamada a la subrutina **“Norte”**, para que el robot se posicione en la trayectoria deseada, que en este caso no es otra que el norte.

Cuando el programa principal ha terminado su inicialización, este entra en el bucle principal, el cual comienza a hacer una medida del dato que le manda el ultrasonido, para conocer a qué distancia esta de un posible objeto. A continuación, hace una lectura del sensor de luminosidad y, si este está en un nivel bajo, enciende los leds, y si no lo está los mantiene apagados.

Una vez ha terminado el SR1 de comprobar la luminosidad, este comprobará si tiene un obstáculo dentro del límite de 30 centímetros que se le ha marcado anteriormente, en caso afirmativo, el robot enciende el led rojo del basicX e incrementa en 1 el valor de una variable llamada **veces**, la cual sirve para que el robot no gire siempre a la izquierda o derecha, basándose este en dicha variable y comprobando, que si esta tiene un valor menor que 2 gire a la izquierda y si es mayor que 2 lo haga hacia la derecha.

Como el programa principal es un bucle infinito, se debe hacer un bucle que resetee la variable **veces** de vez en cuando, ya que si no, llegaría un punto que siempre giraría en cada obstáculo a la derecha. Por lo tanto, una vez se incrementa la variable una unidad, el programa comprueba si esta es mayor de 4, y en caso de ser afirmativo, la pone de nuevo a 0. De esta manera el robot siempre gira 2 veces consecutivas hacia el lado izquierdo y dos veces consecutivas para el derecho.

Como lo que interesa es que el robot esquive completamente el obstáculo, el siguiente paso que hace el programa es un bule que hace que el robot gire hasta que el sensor de ultrasonidos le indique que el obstáculo ha desaparecido. Dicho bucle comprueba primero, gracias a la variable **veces** explicada con anterioridad, si va a evitar el obstáculo girando hacia la izquierda o hacia la derecha, haciendo una llamada a la subrutina **“izquierda”** o **“derecha”** dependiendo del giro.

Hay un punto que hay que destacar de este bucle, y es la variable **DiCam**, cuyo nombre es la abreviatura de **“Dirección de la cámara”**. Gracias a ella, el programa una vez tenga que girar la cámara para la dirección donde este el obstáculo, puede saber si este está en el lado izquierdo o derecho del robot, dependiendo de si la variable tiene un valor de 0 o 1. Por último, el bucle vuelve a tomar la medida de la distancia para saber si el robot ya ha evitado el obstáculo o no.

Una vez el robot ha salido del bucle, este hace una llamada a la subrutina **“DireccionCamara”** pasándole la variable **Dicam** como parámetro. Dicha subrutina se encarga de girar la cámara hacia la dirección donde se encuentra el obstáculo, encender la cámara para tomar una imagen de este, apagarla, y volverla a colocar en su posición central.

Al salir de la subrutina se resetea la variable **Dicam**, colocándola a 0 para que el programa no cometa el error de girar la cámara siempre hacia la misma dirección. Por último, el programa hace una llamada a la subrutina **“Adelante”** para que el robot avance una distancia prudencial del obstáculo y vuelve a hacer una llamada a la subrutina **“Norte”**, para que gracias a ella, el robot retome el rumbo y apaga el led rojo indicando que el obstáculo ha sido sorteado.

Como dependiendo de la clase del objeto, el sensor de ultrasonidos puede fallar, el programa hace utilidad de los parachoques del SR1 y del sensor de inclinación, por si al colisionar con el objeto, este no tiene la suficiente altura como para activar los parachoques y desnivela el robot.

En primer lugar el programa comprueba si el parachoques izquierdo está pulsado, en caso afirmativo comprueba también si el derecho esta pulsado también, si también se cumple esta situación, el robot llama a la subrutina **"Atrás"** y, una vez este termina de retroceder, llama a la subrutina **"GiroD"** para sortear el obstáculo girando hacia la derecha.

Si el programa detecta que el objeto solo ha colisionado en el parachoques izquierdo, el programa llama a la subrutina **"GiroD"** únicamente, de la misma manera que si detecta que el único parachoques golpeado es el derecho llama a la subrutina **"GiroI"** para girar solamente hacia la izquierda.

Después de comprobar el estado de los parachoques, el programa, como se ha mencionado antes, hace una lectura del sensor de inclinación mediante la instrucción **If Getpin(Inc) = 0**, de esta manera se sabe si el robot esta en desnivel. Si lo está, el robot hace una llamada a la subrutina **"Beep_Beep"** para hacer un pitido de alerta. Acto seguido, llama a la subrutina **"Atras"** para retroceder y a la de **"GiroD"** para sortear el obstáculo. Como es normal, lo último que hace el robot es que, si ninguno de sus sensores ha sido activado, señal inequívoca de que no hay ningún obstáculo, el robot avanza de forma continua hacia delante.

5.2.3 Explicación de las subrutinas

Una vez explicadas las funciones del programa principal, se enumeran las diferentes subrutinas de las que está formado el programa, para que la comprensión de este sea total, y mostrar el programa en su totalidad.

Las Subrutinas más importantes **son "DireccionCamara"**, **"Norte"** , **"Dir"**, **"Adelante"**, **"Derecha"**, **"Izquierda"**, **"Pulsos"**, **"Distancia"**, **"GiroD"**, **"GiroI"**, **"Atras"**, **"ServoStop"**, **"Beep_Beep"** y **"Beep"**, las cuales son descritas una a una en los siguientes apartados expuestos a continuación:

5.2.3.1 La subrutina "DireccionCamara"

La subrutina **"DireccionCamara"** es la que se encarga de girar la cámara hacia el lugar donde está el obstáculo y tomar un pequeño video de este, para una vez filmado, volver a colocar la cámara en su posición central.

Para ello se vale de la variable **DiCam**, que es el parámetro que le pasa el programa principal para decirle si ha de girar la cámara hacia la izquierda o hacia la derecha. Esto último es la primera función que hace el programa, y como lo único que varia es la dirección hacia donde gira la cámara, se hace una explicación global de la subrutina en ambas direcciones.

Una vez la subrutina ha comprobado el lado hacia donde ha de girar la cámara, este recupera el valor máximo o mínimo (dependiendo de la dirección a girar la cámara) del Servo4, el cual es el que se encarga de girar la cámara hacia los lados, y efectúa un giro de 90º hacia el lugar donde está el obstáculo. Acto seguido, el programa hace un **Call PutPin(Evc, 1)**, que es la instrucción que enciende la cámara, y comienza un bucle for, cuya finalidad es hacer un retardo para que la cámara permanezca un periodo de tiempo encendida.

Cuando el retardo ha concluido, se activa la instrucción **Call PutPin(Evc, 0),** la cual se encarga de pagar la cámara, y finaliza recuperando el valor central del servo4 y colocando la cámara en esa posición. Una vez terminados estos pasos se finaliza la subrutina.

5.2.3.2 La subrutina “Norte”

La subrutina **“Norte”** hace uso de la brújula del SR1, previamente calibrada en el proceso de calibración del robot (**Ver apartado 3.5.7**). La subrutina comienza dándole unos valores de aceleración a los servos. Una vez hecho esto, la subrutina hace una llamada a la subrutina **“Dir”**, que es la que se encarga de comprobar la dirección en la que esta posicionado el robot, leyendo el valor que le pasa la brújula en grados y quitándole los decimales para obtener un número entero.

En ese momento, la subrutina, gracias al dato en grados de la dirección, comprueba si este es mayor o menor de 180º. Con ello se logra que, cuando el robot gire hacia el norte, lo haga por el lado más próximo a este, haciendo el menor giro posible.

Una vez que la subrutina ha visto la dirección a la que el robot ha de girar, comienza un bucle que dura hasta que el robot este orientado hacia el norte. Dentro de dicho bucle, se comprueba la dirección constantemente para no pasarse del norte y, una vez el robot esté próximo a este, irán frenándose los servomotores del robot, y a la vez, se irá controlando también que estos no se pasen de su valor máximo y mínimo en ningún momento por precaución efectuando una llamada a la subrutina **“pulsos”**. Una vez el robot ha llegado al norte sale de la subrutina y vuelve al bucle principal.

5.2.3.3 La subrutina “Dir”

La subrutina **“Dir”** es la que se encarga de decir en qué posición se encuentra el robot, leyendo los datos de la brújula en grados y quitándole los decimales para así obtener un número entero y poder usar variables **integer**(variables que usan números enteros).

5.2.3.4 Las subrutinas “Adelante” y “Atras”

Las subrutinas **“Adelante”** y **“Atras”** son las que se encargan de la movilidad del robot hacia delante o hacia atrás, como bien dice el nombre de las subrutinas. Ambas son prácticamente iguales, solo que en la subrutina **“Adelante”** se mandan pulsos a los servos para que las ruedas avancen, y en la de **“Atrás”**, para que retrocedan.

La subrutina comienza poniendo los servos en su posición de parada, y creando un bucle for que determina la distancia que recorre el robot, dicha distancia depende de la variable que domina el bucle.

Hay que especificar que dentro del bucle, también se efectúa una llamada a la subrutina **“Pulsos”**, para comprobar que los servos no salen de su valor máximo y mínimo y de esta manera no se dañen.

5.2.3.5 Las subrutinas “Derecha” e “Izquierda”

Las subrutinas **“Derecha”** e **“Izquierda”** son las que se encargan de girar el robot hacia la derecha y la izquierda como su nombre indican. Ambas funcionan de la misma manera que las subrutinas **“Adelante”** y **“Atras”**, con la única diferencia de que en las subrutinas de giro, los pulsos que se le mandan a los servos dentro del bucle for son opuestos, haciendo que el robot gire sus ruedas de manera opuesta para que este realice giros hacia la izquierda o la derecha, dependiendo de la subrutina.

5.2.3.6 La subrutina “Pulsos”

La subrutina “Pulsos” es la que se encarga de que los servos que mueven el robot, estén dentro de sus límites y que no sobrepasen sus valores máximos y mínimos, y si los sobrepasan, los vuelve a colocar dentro de sus rangos para que no funcionen de manera deficiente.

5.2.3.7 La subrutina “Distancia”

La subrutina “Distancia” es la encargada de comprobar a la distancia que esta un objeto del sensor de ultrasonidos, y de leer los niveles de luz a los que está sometido el sensor lumínico, es esencial tanto para poder evitar las colisiones de los objetos, como para poder encender y apagar los leds del SR1.

5.2.3.8 Las subrutinas “GiroD” y “GiroI”

Las subrutinas **“GiroD”** y **“GiroI”** son las que se encargan de que el robot haga giros hacia la derecha e izquierda respectivamente. Estas subrutinas son prácticamente iguales que la de **“izquierda”** y **“derecha”**, solo que en estas, las subrutinas comprueban si ya se ha efectuado un giro con anterioridad, y en caso de que sea afirmativo, realizan el giro más largo, ya que estas subrutinas son usadas por los parachoques.

Gracias a esta subrutina se puede evitar que el robot se quede clavado en un punto muerto intentando salir de un objeto, realizando giros cortos de izquierda a derecha.

5.2.3.9 La subrutina “ServoStop”

La subrutina “ServoStop” es la que se encarga de leer de la memoria eeprom del SR1 los valores de parada de los servos que se encargan de mover el robot, gracias a esta subrutina se pueden volver a recuperar los valores de parada de los servos siempre que sea necesario.

5.2.3.10 Las subrutinas “Beep_Beep” y “Beep”

Por último las subrutinas **“Beep_Beep”** y **“Beep”** son las que se encargan de hacer una alarma sonora de un tono, en el caso de la subrutina **“Beep”**, y de dos tonos más parpadeos de leds, en la **“Beep_Beep”**.

En primer lugar las subrutinas asignan un número entero a una variable cuyo valor es 2000, que es la frecuencia de resonancia del altavoz que incorpora el SR1, y por lo tanto, es la más ruidosa de todas. A continuación, hace un **Call PutPin(Led, Lon)** para encender los leds del SR1, después efectúa el primer Beep mandándole al altavoz la frecuencia de 2000 y, si es la subrutina de **“Beep_Beep”** repite este paso otra vez para efectuar el segundo tono, por último se apagan los leds del robot y se dale de la subrutina.

5.2.4 Diagrama de bloques del programa

Para una perfecta comprensión del programa, en la figura 5.6 se puede ver el diagrama de bloques del programa principal con todos sus pasos y variantes.

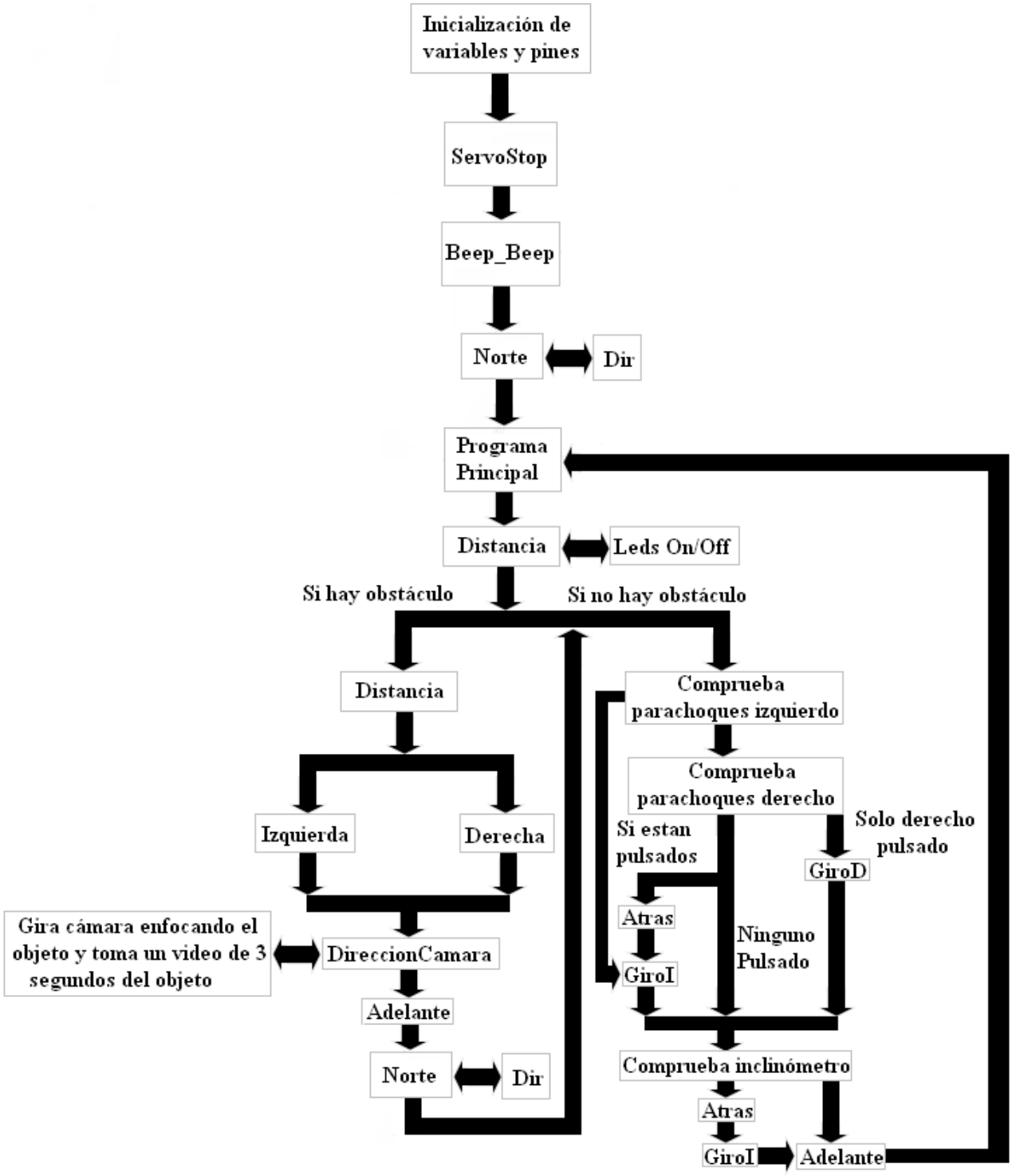


Fig.5.6 *Diagrama de bloques del programa "Dirección Norte"*

5.2.5 Valoración del programa “Dirección Norte”

La valoración del programa “Dirección Norte” es excelente, estando el programa pulido por completo de fallos.

La primera versión del programa consistía en que el robot calculara el norte y acto seguido avanzara hasta chocar con un obstáculo y en ese momento girar gracias a los parachoques. La segunda versión era una mejora de la primera versión con la implementación del uso del sensor de ultrasonidos y que una vez el robot esquivara el obstáculo volviera a dirigirse al norte. En esta versión, el principal problema era que el robot, una vez había esquivado un obstáculo, si el segundo estaba cerca no le daba tiempo de esquivarlo con el sensor de ultrasonidos y tenía que hacer uso de los parachoques. Dicho error fue subsanado en la versión tercera del programa, modificando el código para que los ultrasonidos respondieran de forma más rápida. En su cuarta versión, se le añadió al código del programa la opción de girar la cámara cada vez que el robot encontrara un obstáculo, siendo esta opción mejorada en la última versión del código con la variable **DiCam**, la cual permitía al robot girar la cámara en la dirección donde se encontraba el obstáculo.

En su versión final, el robot ha sido probado en varios tipos de circuitos tanto en el interior de una vivienda con espacios reducidos como en el exterior, siendo los resultados excelentes en ambos casos. Los videos de las pruebas pueden verse en el CD adjunto a la memoria en la carpeta videos, donde se encuentran videos realizados en el interior de un recinto y videos realizados en el exterior. También se encuentra en la carpeta de programas, el código del programa bien explicado y detallado en formato pdf para aclarar el programa realizado.

Como conclusión final, solo queda comentar que el robot, en este programa, hace uso de casi todos los sensores y dispositivos que tiene de forma optima y eficiente, siendo el programa más completo de todos los realizados.

5.3 Modificación programa EXPLORER

En este apartado se da la solución a los problemas que se encuentran en el programa del PC el robot. Los problemas que surgen son los siguientes:

- Cuando se le da la orden de girar el robot a la derecha, en vez de girar a la derecha gira a la izquierda y viceversa.

Para éste problema al hacer la acción inversa cuando se aprietan los botones, tiene fácil solución, ésta se ve en la figura 5.7. La solución es invertir las subrutinas, de esta manera se consigue que gire en el sentido que se quiere.

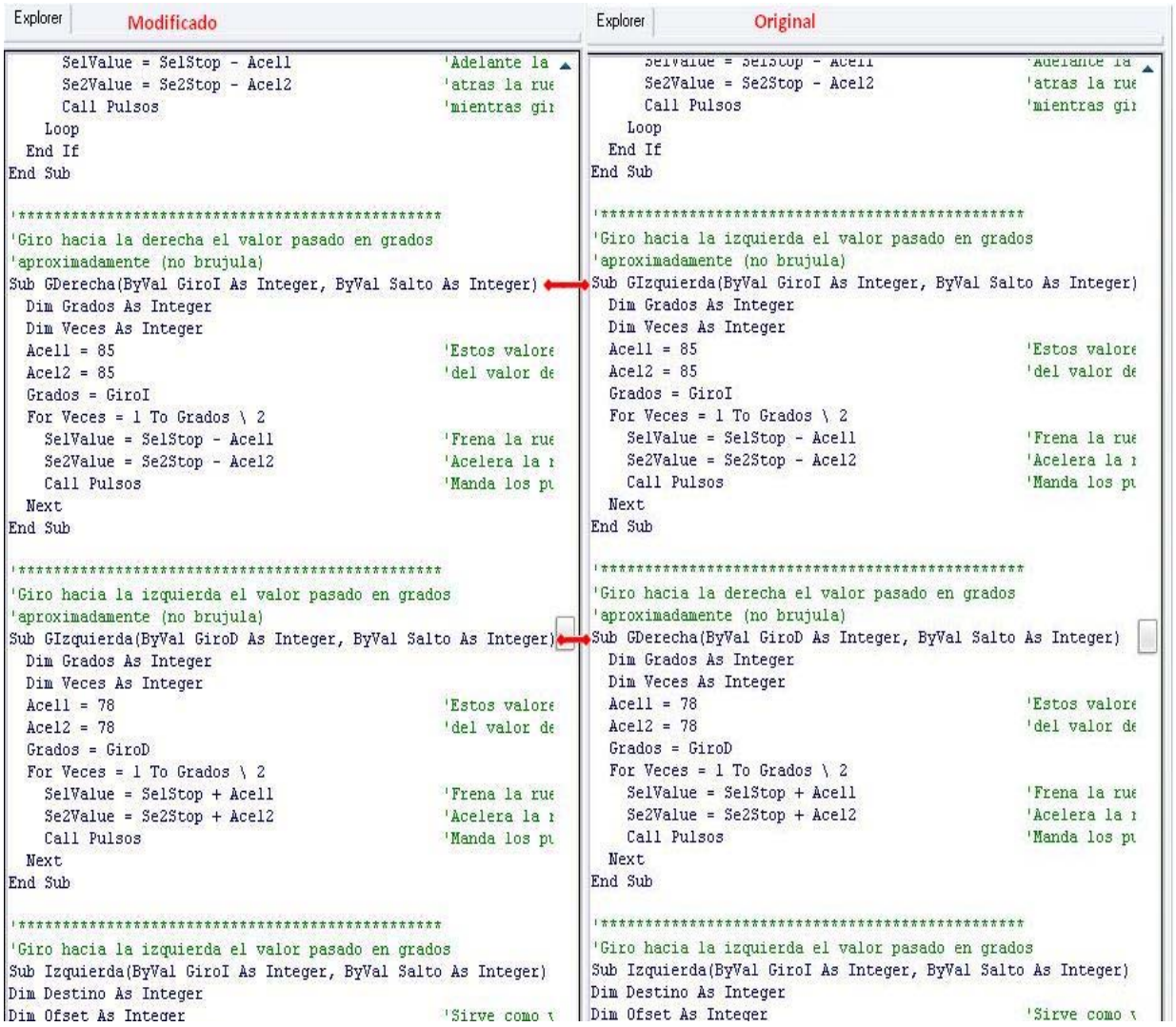
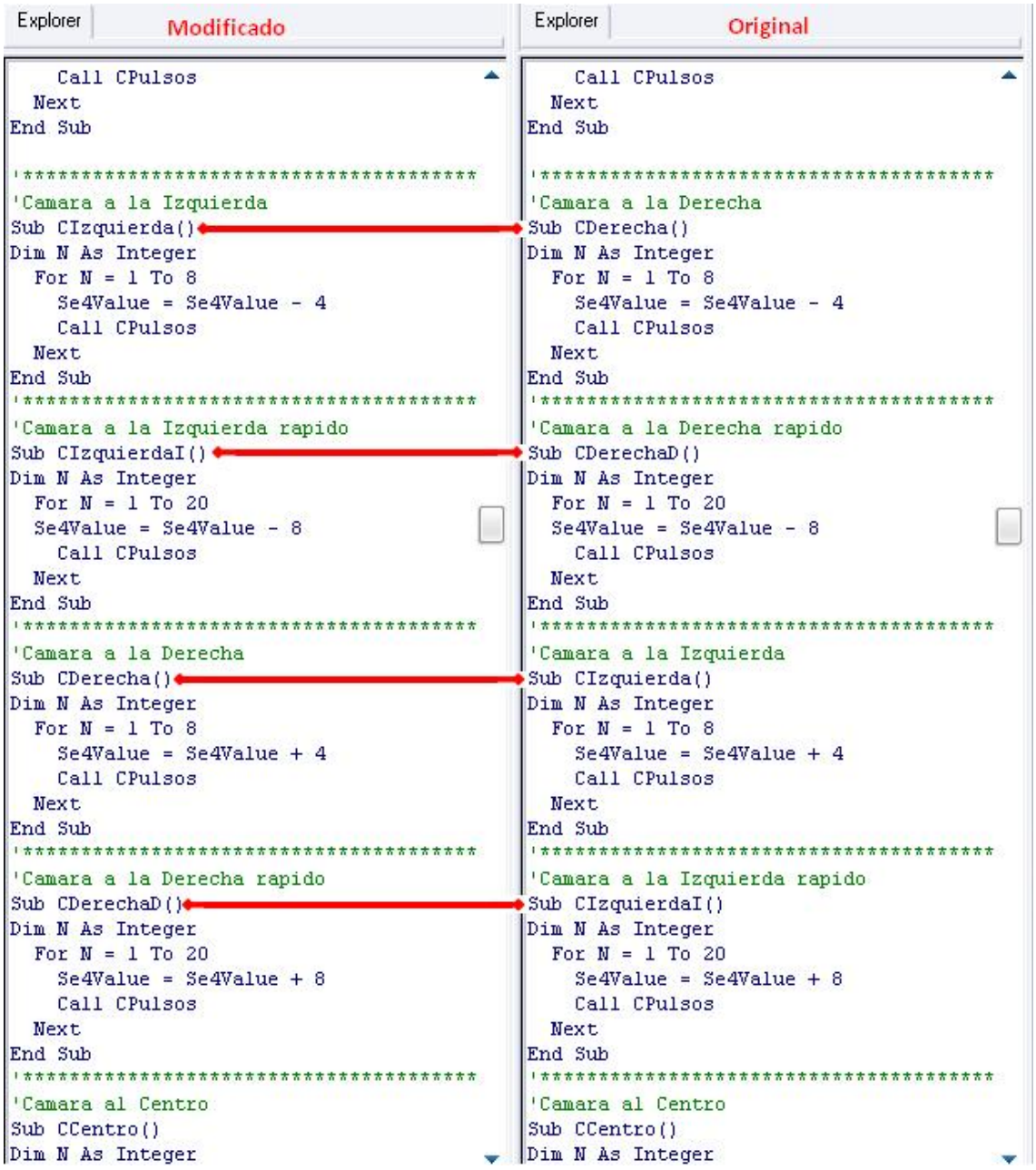


Fig. 5.7 *Modificación del código*

- Cuando se le da la orden de girar la cámara a la derecha, sucede lo mismo, en vez de girar a la derecha gira a la izquierda y viceversa.

Ésta solución es la misma a la anterior ya que es el mismo problema, hace la acción inversa del botón pulsado. La solución de ésta se ve reflejada en la figura 5.8.




```
    Call CPulsos
  Next
End Sub

'**************************************
'Camara a la Izquierda
Sub CIzquierda()
Dim N As Integer
  For N = 1 To 8
    Se4Value = Se4Value - 4
    Call CPulsos
  Next
End Sub
'**************************************
'Camara a la Izquierda rapido
Sub CIzquierdaI()
Dim N As Integer
  For N = 1 To 20
  Se4Value = Se4Value - 8
    Call CPulsos
  Next
End Sub
'**************************************
'Camara a la Derecha
Sub CDerecha()
Dim N As Integer
  For N = 1 To 8
    Se4Value = Se4Value + 4
    Call CPulsos
  Next
End Sub
'**************************************
'Camara a la Derecha rapido
Sub CDerechaD()
Dim N As Integer
  For N = 1 To 20
    Se4Value = Se4Value + 8
    Call CPulsos
  Next
End Sub
'**************************************
'Camara al Centro
Sub CCentro()
Dim N As Integer
```

```
    Call CPulsos
  Next
End Sub

'**************************************
'Camara a la Derecha
Sub CDerecha()
Dim N As Integer
  For N = 1 To 8
    Se4Value = Se4Value - 4
    Call CPulsos
  Next
End Sub
'**************************************
'Camara a la Derecha rapido
Sub CDerechaD()
Dim N As Integer
  For N = 1 To 20
  Se4Value = Se4Value - 8
    Call CPulsos
  Next
End Sub
'**************************************
'Camara a la Izquierda
Sub CIzquierda()
Dim N As Integer
  For N = 1 To 8
    Se4Value = Se4Value + 4
    Call CPulsos
  Next
End Sub
'**************************************
'Camara a la Izquierda rapido
Sub CIzquierdaI()
Dim N As Integer
  For N = 1 To 20
    Se4Value = Se4Value + 8
    Call CPulsos
  Next
End Sub
'**************************************
'Camara al Centro
Sub CCentro()
Dim N As Integer
```

Fig. 5.8 *Modificación del código*

- Cuando se intenta girar el robot hacia la dirección de la cámara, el robot gira hacia donde la cámara está enfocada, pero la cámara no vuelve a su posición central.

La solución a éste problema es colocar la subrutina **Call CCentro**. De esta manera se consigue que después de que el robot gire hacia la posición donde está enfocada la cámara, la cámara vuelva a su posición del centro. El código modificado se ve en la figura 5.9.

Explorer **Modificado**

```
'*************************************
'Gira el robot hacia la posicion de la c
Sub Camara()
  Dim Giro As Integer
  Dim Paso As Integer
  Paso = (Se4Max - Se4Min) \ 180
  If Se4Value = Se4Med Then
    Exit Sub
  End If
  If Se4Value > Se4Med Then
    Giro = (Se4Value - Se4Med) \ Paso
    Paso = Paso * 3
    Call Izquierda(Giro, Paso)
  Else
    Giro = (Se4Med - Se4Value) \ Paso
    Paso = Paso * 3
    Call Derecha(Giro, Paso)
  End If
  Call CCentro
End Sub

'*************************************
'Giro hacia el norte
'Hace un giro hacia el norte por el lado
Sub Norte()
  Acel1 = 80
  Acel2 = 80
  Call Dir
  If DirACT < 180 Then
    Do
      Call Dir
      If DirACT > 345 Then
        Acel1 = 40
        Acel2 = 40
      End If
```

Explorer **Original**

```
'*************************************
'Gira el robot hacia la posicion de la c
Sub Camara()
  Dim Giro As Integer
  Dim Paso As Integer
  Paso = (Se4Max - Se4Min) \ 180
  If Se4Value = Se4Med Then
    Exit Sub
  End If
  If Se4Value > Se4Med Then
    Giro = (Se4Value - Se4Med) \ Paso
    Paso = Paso * 3
    Call Izquierda(Giro, Paso)
  Else
    Giro = (Se4Med - Se4Value) \ Paso
    Paso = Paso * 3
    Call Derecha(Giro, Paso)
  End If
End Sub

'*************************************
'Giro hacia el norte
'Hace un giro hacia el norte por el lado
Sub Norte()
  Acel1 = 80
  Acel2 = 80
  Call Dir
  If DirACT > 180 Then
    Do
      Call Dir
      If DirACT > 345 Then
        Acel1 = 40
        Acel2 = 40
      End If
```

Fig. 5.9 *Modificación del código*

- Cuando se quiere girar hacia el norte, después de calibrar bien la brújula como se explica en el apartado 3.6, el robot solo gira por el camino más largo, nunca hace el giro más óptimo.

Como se ve en la figura 5.10, la solución a este problema es la de variar una comparación, en la que en el programa original nunca gira por el camino más corto. Una vez cambiada la comparación el robot busca el camino más corto para girar hacia el norte.

Explorer **Modificado**

```
'****************************************
'Giro hacia el norte
'Hace un giro hacia el norte por el
'lado mas corto
Sub Norte()
  Acel1 = 80
  Acel2 = 80
  Call Dir
  If DirACT < 180 Then
    Do
      Call Dir
      If DirACT > 345 Then
        Acel1 = 40
        Acel2 = 40
      End If
      If DirACT < 10 Then
        Exit Do
      End If
      SelValue = SelStop + Acel1
      Se2Value = Se2Stop + Acel2
      Call Pulsos
    Loop
  Else
    Do
      Call Dir
      If DirACT < 15 Then
        Acel1 = 40
        Acel2 = 40
      End If
      If DirACT > 350 Then
        Exit Do
      End If
      SelValue = SelStop - Acel1
      Se2Value = Se2Stop - Acel2
      Call Pulsos
    Loop
  End If
End Sub

'****************************************
'Giro hacia la derecha el valor pasado e
'aproximadamente (no brujula)
Sub GDerecha(ByVal GiroI As Integer, ByV
```

Explorer **Original**

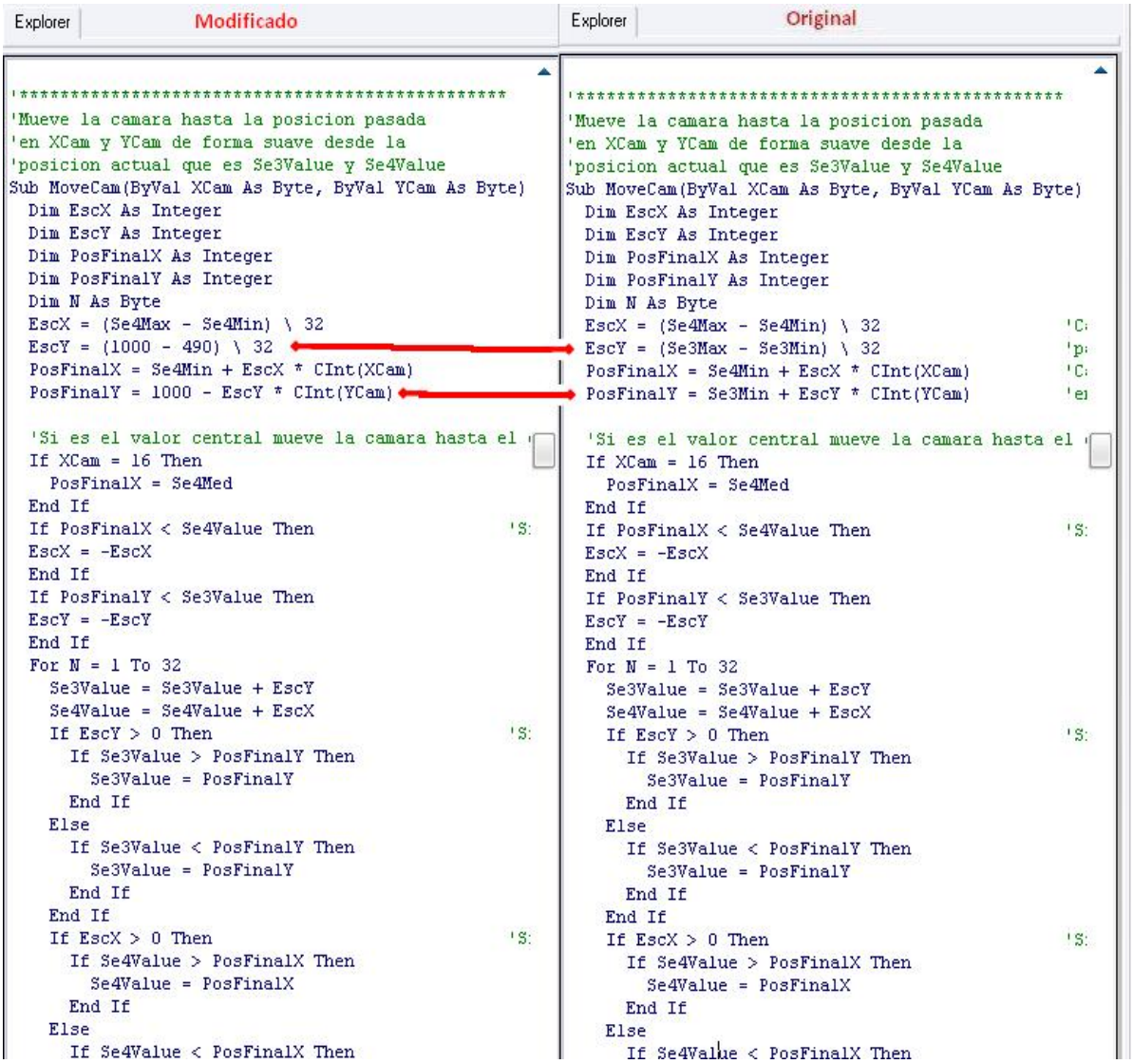
```
'****************************************
'Giro hacia el norte
'Hace un giro hacia el norte por el
'lado mas corto
Sub Norte()
  Acel1 = 80
  Acel2 = 80
  Call Dir
  If DirACT > 180 Then
    Do
      Call Dir
      If DirACT > 345 Then
        Acel1 = 40
        Acel2 = 40
      End If
      If DirACT < 10 Then
        Exit Do
      End If
      SelValue = SelStop + Acel1
      Se2Value = Se2Stop + Acel2
      Call Pulsos
    Loop
  Else
    Do
      Call Dir
      If DirACT < 15 Then
        Acel1 = 40
        Acel2 = 40
      End If
      If DirACT > 350 Then
        Exit Do
      End If
      SelValue = SelStop - Acel1
      Se2Value = Se2Stop - Acel2
      Call Pulsos
    Loop
  End If
End Sub

'****************************************
'Giro hacia la izquierda el valor pasado
Sub GIzquierda(ByVal GiroI As Integer, B
```

Fig. 5.10 *Modificación del código*

-Cuando se aprieta el botón de hacer subir la cámara, ésta sube. Pero al intentar hacer que baje la cámara, la cámara no baja.

Este problema se produce porque se tiene que modificar la parte del calibrado de la cámara, como se ve en el apartado 3.6. Se usa el programa de calibrado de la cámara para saber los valores máximos y mínimos que da el servomotor que mueve la cámara en vertical. Una vez se obtienen dichos valores se ponen directamente en el código, ya que el programa **Explorer** los detecta erróneamente. En la figura 5.11 se muestra el cambio producido para solucionar este conflicto.




```
'************************************************
'Mueve la camara hasta la posicion pasada
'en XCam y YCam de forma suave desde la
'posicion actual que es Se3Value y Se4Value
Sub MoveCam(ByVal XCam As Byte, ByVal YCam As Byte)
  Dim EscX As Integer
  Dim EscY As Integer
  Dim PosFinalX As Integer
  Dim PosFinalY As Integer
  Dim N As Byte
  EscX = (Se4Max - Se4Min) \ 32
  EscY = (1000 - 490) \ 32
  PosFinalX = Se4Min + EscX * CInt(XCam)
  PosFinalY = 1000 - EscY * CInt(YCam)

  'Si es el valor central mueve la camara hasta el
  If XCam = 16 Then
    PosFinalX = Se4Med
  End If
  If PosFinalX < Se4Value Then                'S
  EscX = -EscX
  End If
  If PosFinalY < Se3Value Then
  EscY = -EscY
  End If
  For N = 1 To 32
    Se3Value = Se3Value + EscY
    Se4Value = Se4Value + EscX
    If EscY > 0 Then                          'S
      If Se3Value > PosFinalY Then
        Se3Value = PosFinalY
      End If
    Else
      If Se3Value < PosFinalY Then
        Se3Value = PosFinalY
      End If
    End If
    If EscX > 0 Then                          'S
      If Se4Value > PosFinalX Then
        Se4Value = PosFinalX
      End If
    Else
      If Se4Value < PosFinalX Then
```

```
'************************************************
'Mueve la camara hasta la posicion pasada
'en XCam y YCam de forma suave desde la
'posicion actual que es Se3Value y Se4Value
Sub MoveCam(ByVal XCam As Byte, ByVal YCam As Byte)
  Dim EscX As Integer
  Dim EscY As Integer
  Dim PosFinalX As Integer
  Dim PosFinalY As Integer
  Dim N As Byte
  EscX = (Se4Max - Se4Min) \ 32               'C
  EscY = (Se3Max - Se3Min) \ 32               'p
  PosFinalX = Se4Min + EscX * CInt(XCam)      'C
  PosFinalY = Se3Min + EscY * CInt(YCam)      'e

  'Si es el valor central mueve la camara hasta el
  If XCam = 16 Then
    PosFinalX = Se4Med
  End If
  If PosFinalX < Se4Value Then                'S
  EscX = -EscX
  End If
  If PosFinalY < Se3Value Then
  EscY = -EscY
  End If
  For N = 1 To 32
    Se3Value = Se3Value + EscY
    Se4Value = Se4Value + EscX
    If EscY > 0 Then                          'S
      If Se3Value > PosFinalY Then
        Se3Value = PosFinalY
      End If
    Else
      If Se3Value < PosFinalY Then
        Se3Value = PosFinalY
      End If
    End If
    If EscX > 0 Then                          'S
      If Se4Value > PosFinalX Then
        Se4Value = PosFinalX
      End If
    Else
      If Se4Value < PosFinalX Then
```

Fig. 5.11 *Modificación del código*

- A la solución de este problema se produce otro problema. El error producido es que cuando se quiere que la cámara se mueva hacia arriba, la cámara se mueve hacia abajo y viceversa.

- Cuando se quiere hacer el barrido con el sonar de ultrasonidos, siempre hace el barrido bajo por culpa de que el servomotor vertical de la cámara no funciona correctamente.

La solución a estos dos problemas se produce prácticamente a la vez. Los dos problemas van relacionados con la posición vertical de la cámara y por lo tanto, todo tiene cierta relación.

La figura 5.12 muestra la solución a los dos problemas mencionados.

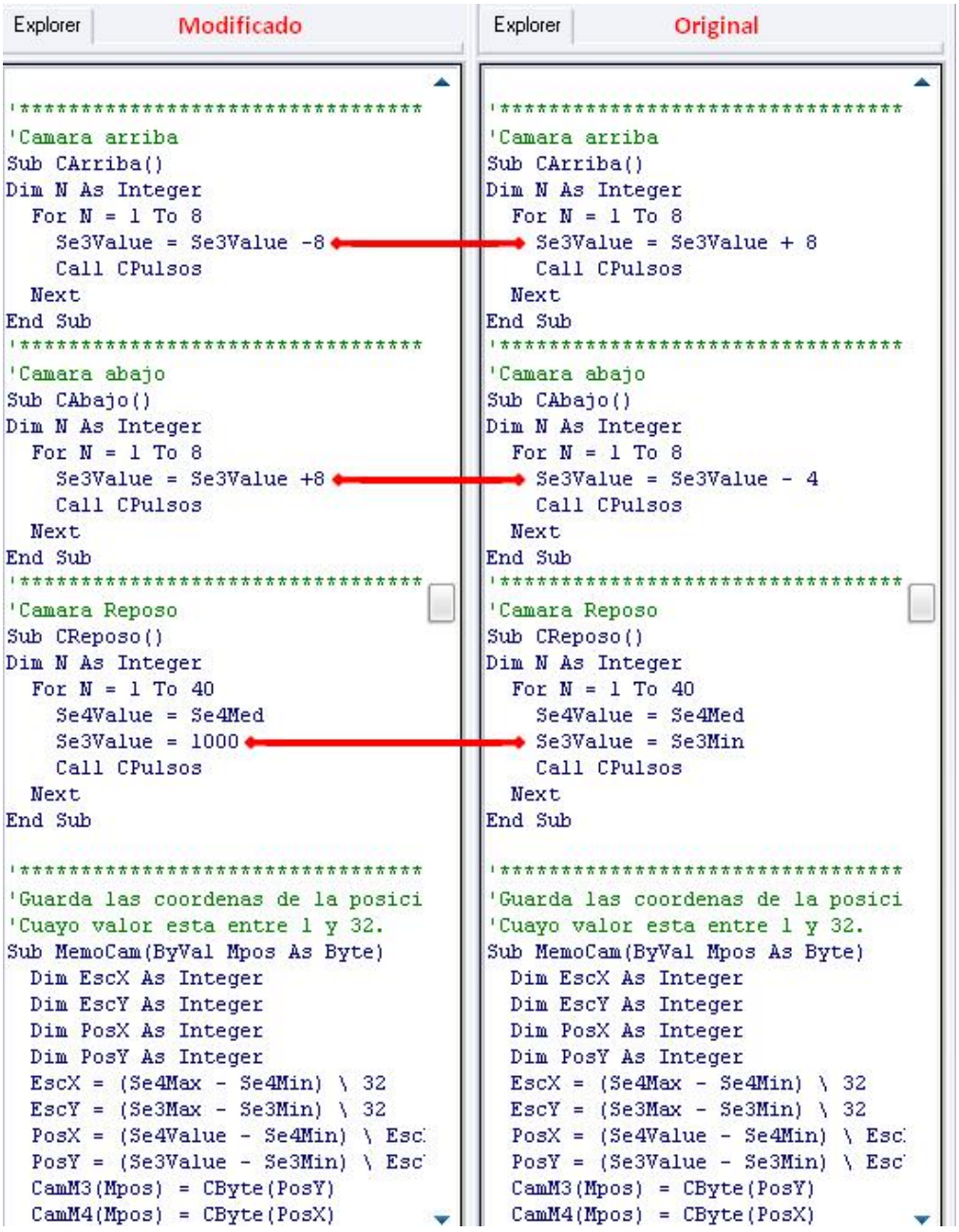


Fig. 5.12 *Modificación del código*

6. Conclusiones

6.1 Conclusiones del montaje

Para comenzar se comentan las conclusiones obtenidas en el montaje de la placa y el chasis. En primer lugar hay que destacar la placa del robot, ya que está perfectamente insolada y con todas sus conexiones estañadas, cosa que facilita la soldadura de los componentes.

Muchos puntos de conexión requieren una cierta habilidad soldando, ya que muchos de los dispositivos son muy frágiles, y si el usuario se pasa calentando el punto de soldadura, puede llegar a estropear más de un sensor o el microprocesador BasicX24. También son de mucha utilidad las explicaciones del manual, siendo estas detallistas y perfectamente explicativas, y tener a disposición un soldador de precisión y una estación de soldadura.

En cuanto a la parte del chasis, hay que destacar la fragilidad del material y la no inclusión de agujeros en las piezas para poder atornillarlas entre sí. Dicha ausencia de agujeros tiene que ser solucionada con un taladro y una broca muy fina, con los que, con mucho cuidado, se hacen los agujeros en las piezas una vez se han marcado los lugares a taladrar.

Un punto negativo es que, a diferencia de la parte de soldadura, en la parte de chasis el manual no es muy aclaratorio, omitiendo a veces los tornillos necesarios a usar y necesitando, en muchas ocasiones, un pie de rey para asegurar que se está usando el tornillo correcto. Por último comentar que muchas de las piezas del chasis tienen que ser limadas para rebajarse, como en el caso de los servos, ya que rozan demasiado las ruedas con el chasis, perdiendo el robot mucha potencia, o en la pieza donde fue colocado por primera vez el sensor de ultrasonidos, que tiene que ser rebajada más de la mitad para que coincida en su posición.

La parte de modificación de los servos está correctamente explicada en el manual y no es difícil de realizar, aunque si laboriosa y poco aprovechable, por lo que habría sido un acierto que el fabricante del robot hubiera puesto a disposición del usuario unos servos de 360º de rotación.

6.2 Conclusiones del Entorno de programación

Las conclusiones que se sacan de la programación son un poco dispares. Lo primero a comentar es el programa BasicX, siendo este de rápida comprensión y de fácil programación. Los programas creados con BasicX son fáciles de entender, predominando los bucles, las condiciones y las subrutinas. La simplicidad del programa es algo a agradecer, ya que cuando algo no funciona del programa el usuario intuye donde puede estar el problema y, de esta forma, darle solución.

La nota discordante viene dada por el entorno de programación, ya que muchas veces que se quiere crear un programa de cero, hay que tener mucho cuidado de crear un programa base, ya que si no, muchas veces el programa hereda el programa base que estuviera abierto con anterioridad sobrescribiéndolo. De esta manera se pueden perder muchos programas si no se tiene cuidado con este matiz.

El segundo punto negativo es que el editor de programas carece de muchas opciones de las que si disponen otros editores de otros programas, como Java con Oracle. Haciendo que en repetidas ocasiones, poder ordenar correctamente un bucle o el programa principal sea demasiado laborioso. De esta manera hace que el usuario, o usuarios posteriores que quieran mejorar los programas, puedan tener dificultades para entenderlos.

6.3 Conclusiones de los programas específicos y de calibración

Los programas que ofrece el fabricante del robot para realizar pruebas del robot suelen ser todos correctos, habiendo algunos en los que se tienen que mejorar pequeños detalles para que estos funcionaran mejor, o para sacar más partido a los sensores del robot. El único que llega a dar problemas graves es el "**Explorer"**, ya que cuando se prueba por primera vez el 50% de sus aplicaciones no funcionan, teniendo que hacer una gran remodelación en el programa hasta poder conseguir que funcione correctamente.

En cuanto a los programas de calibración, son de muy fácil manejo, haciendo que la calibración del robot resulte no muy complicada y, además de calibrar los sensores, el usuario aprende cómo funcionan estos, pudiendo sacar de esta manera el mejor partido de los sensores y servos en programas propios.

El único programa que no funciona correctamente fue el programa **"Camset"**, que se encarga de calibrar los servos de la cámara. El programa, una vez volcado, no hace que los servos de la cámara actúen de forma correcta, impidiendo así su calibración, el problema es solucionado modificando el código del programa, tal y como puede verse en el **apartado 3.5.2.1** de esta memoria.

6.4 Conclusiones de los programas realizados

Esta es la parte que, sin duda, cuesta más de realizar. Los programas propios constan del programa **"Control Remoto SonyIR"**, el programa **"Dirección Norte"** y la modificación del programa **"Explorer"**.

El programa **"Control Remoto SonyIR"** está realizado con mucho cuidado, implementando poco a poco el programa principal y cada subrutina a una tecla del mando a distancia. La parte más dura es la de realizar los giros de 15, 45 y 90 grados, ya que en cada superficie y, dependiendo del estado de la batería del robot, el robot varia su giro, por lo que se tienen que realizar varias pruebas hasta ajustar el valor de los giros. También cabe destacar los dos métodos realizados para encender la cámara y los leds usando un solo botón en cada caso, y la buena respuesta del receptor de ultrasonidos, captando este las señales a más de 4 y 5 metros del usuario.

El programa **"Dirección Norte"** es el más complicado de los tres programas. Primero se comienza a implementar que el robot cada vez que encuentre un obstáculo gire y vuelva a posicionarse hacia el norte, siendo insuficiente para sortear algunos obstáculos. La siguiente mejora es adaptarle el sistema para que esquive el obstáculo de una forma total y, después de eso, volver a posicionarse hacia el Norte. Dicho paso costó de implementar correctamente 2 semanas, ya que se encontraron muchos problemas que impedían que el robot funcionara de forma correcta. Ya por último se implementa la filmación del objeto antes de sortearlo, pudiendo visualizarlo en una televisión o ordenador gracias al receptor de radio que venía de seria con la cámara.

Este programa es, sin lugar a dudas, con el que más se ha llegado a comprender el programa BasicX y el que más explota los recursos del SR1.

El programa **"Explorer"** ha sido modificado de una forma radical para que funcionaran todas sus opciones. En esta parte es difícil localizar los errores causantes del mal funcionamiento del programa y ponerles solución, sobre todo con lo que respecta a los servos de la cámara.

6.5 Conclusiones finales

Todo el proyecto en si sirve para aprender a programar en BasicX y a usar todos los sensores de los que dispone el robot SR1, siendo la programación muy amena e interesante para una carrera de telecomunicaciones. También son provechosos los distintos fallos que incluyen los programas del robot, ya que se ha profundizado en la localización y resolución de errores, algo que ayuda mucho a coger facilidad programando.

En cuanto a los sensores, se profundiza mucho en el sensor de ultrasonidos y en la brújula del robot, aportando dichos sensores nuevos conocimientos no vistos en la carrera y nuevas aplicaciones de esta en el mundo de la robótica.

7. Referencias

[1] Superrobotica, “Manual de instrucciones SR1”

[2] http://www.superrobotica.com , página oficial del Robot SR1

[3] http://www.areasx.it/index.php?d=1&id=8099, datasheet del basicX24

[4] http://www.robot-electronics.co.uk/html/srf08tech.shtml, datasheet sensor de ultrasonidos (SRF08)

[5] http://www.alldatasheet.com/datasheet-pdf/pdf/58548/dallas/ds1820.html, datasheet sensor de temperatura (DS1820)

[6] http://www.alldatasheet.net/datasheet-pdf/pdf/83166/etc/cmps03.html, datasheet sensor brújula digital (CMPS03)

[7] http://www.robotshop.ca/pdf/hitec-hs-425bb-servo-specsheet.pdf, datasheet servomotor (HS-422)

[8] http://www.lynxmotion.com/images/data/tra-v5.pdf, datasheet seguidor de líneas

[9] http://www.superrobotica.com/s320165.htm, datasheet sensor de inclinación

[10] http://www.kevin.org/frc/2004/tsop4840.pdf, datasheet sensor de infrarrojos